POINT SET REGISTRATION VIA PARTICLE FLOW FILTERING

by
Stephen R. Porter

A thesis submitted to Johns Hopkins University in conformity with the requirements for

the degree of Master of Science

Baltimore, Maryland
December 2020

Abstract

This thesis project presents an innovative registration algorithm using the particle flow filter. This is the first known approach to image registration using the particle flow filter. The particle flow filter is a Bayesian filter that uses particles to represent probability densities. Others have approached image registration as a Bayesian filtering problem [1] [2] [3]; however, none have used the particle flow filter. The particle flow filter is not constrained to the highly restrictive unimodal, linear, and Gaussian assumptions of many Bayesian filters such as the Kalman filter. The particle flow filter works for any probability density function. Additionally, the particle flow filter is computationally more efficient than other multimodal filters such as the better-known particle filter. Unlike the particle filter, the particle flow filter does not require particle resampling or importance weight updates. Rather, the proposal density is formed by flowing the *a priori* probability density to the *a posteriori* using the Fokker-Planck equation. Moreover, the particle flow filter is more parallelizable than the particle filter. Regarding image registration, the particle flow filter is more robust to noise and outliers than other methods.

The particle flow filter algorithms were implemented in MATLAB for both 2D and 3D rigid body point-set registration. Additionally, the particle filter method proposed by [1] and iterative closest point algorithms were implemented for comparison. All three registration techniques were tested with a high degree of initial misalignment and noise. For the same alignment accuracy, the new particle flow filter algorithms were 244% faster than the particle filter for certain challenging problems. For the same alignment time, the particle flow filter reduced misalignment by as much as 35% compared to the particle filter. The particle flow filter achieved 100% alignment with enough particles, and reduced misalignment by as much as 75% over that of iterative closest point. These results demonstrate that image registration via the particle flow filter significantly outperforms the particle filter and iterative closest point algorithms in the presence of noise and a high degree of initial misalignment. Future areas of research for particle flow filter image registration include deformable registration and GPU parallelization.

Primary Reader and Advisor: Robert Fry, Johns Hopkins University

Secondary Reader: Cleon Davis, PhD, Johns Hopkins University

Secondary Reader: Fred Daum, Raytheon Technologies

Acknowledgments

I am very grateful to my thesis committee for their help and guidance in the development of this thesis project. In particular, I am thankful for Robert Fry's instruction over the past three years in digital signal processing, probability statistics and stochastic processes, and many other topics such as Bayesian signal processing which was investigated further in this report. Additionally, I am thankful for Fred Daum, the inventor of the particle flow filter, for introducing me to this filtering method and spending a considerable amount of time answering questions as I tried to understand the nuances of this fascinating Bayesian filtering method. His work with the particle flow filter has inspired me, and I am certain it will inspire others in finding many new and innovative applications of the particle flow filter. Finally, I am thankful for Dr. Cleon Davis and the entire Electrical and Computer Engineering leadership team. Their guidance not only contributed to a successful thesis project, but also contributed to my completion of the Master of Science in Electrical Engineering degree from Johns Hopkins Whiting School of Engineering.

Dedication

This thesis is dedicated to my wife and children for their love, patience, and encouragement these past three years as I tirelessly labored in professional and academic study.

Contents

List of Tables

List of Figures

1 Introduction

Image registration is an important function for many applications, which include medical imaging, autonomous driving, facial recognition, artificial intelligence, machine learning, target discrimination and identification, and synthetic aperture radar. As application areas increase, research continues to develop new and more accurate image registration techniques. This project provides one such new method of image registration. This project presents an innovative approach to image registration using the particle flow filter (PFF). The particle flow filter is a Bayesian filter that uses particles to represent probability densities. This is the first known approach to image registration using the PFF. The particle flow filter is closely related to the better-known particle filter (PF); however, the PFF is computationally more efficient than the PF. Image registration fits within the Bayesian filtering process, so it is ideally suited for the PFF.

This paper is divided into 7 sections. Sections 1 and 7 are the introduction and conclusion, respectively. Sections 2 – 6 contain the main discussion and are briefly summarized below:

Section 2 provides an overview of image registration. An understanding of several topics within image registration is needed to appreciate the registration approach presented in this paper. This section begins with an overview of image registration, and concludes with the iterative closest point (ICP) algorithm. ICP is the underlying registration algorithm guided by the PFF proposed in this paper. Image registration takes two or more images and aligns them spatially. Registration aims to find the optimal transformation to align 2D or 3D images by maximizing an objective function that measures the degree of similarity. Image registration transformations are classified as either rigid, affine, or elastic.

Section 3 provides an overview of the particle filter (PF). This section begins by briefly explaining Bayes Law, the basis of Bayesian filtering. The PF uses particles (point masses) and weights (importance) to represent the probability densities for Bayesian filtering. Due to its similarities with the Kalman filter, a quick tutorial of the Kalman filter is provided to more easily explain the particle filter and particle flow filter. Moreover, the advantages of the PF are readily apparent in light of the Kalman filter's highly restrictive unimodal, linear, and Gaussian assumptions. The PF does not make these assumptions. This section continues with several inherit problems of the particle filter, namely particle degeneracy and particle impoverishment. Particle degeneracy occurs when the variance of the importance weights increases with time and when the weights approach zero. Particle impoverishment is the collapse of particles to a single point. Finally, this section concludes with the algorithmic steps of the particle filter.

Section 4 describes the particle flow filter. The PFF was designed to avoid the shortfalls of the PF. Particle weight updates, resampling, and regularization are not required. The reduction of algorithmic steps leads to an increase in computational efficiency. Instead, the PFF flows the probability densities from the *a priori* to the *a posteriori* via the Fokker-Planck equation. This migration of densities is known as *homotopy*, and is modeled by the Fokker-Planck partial differential equation. There are several PFF variants depending on the assumptions and methods used to solve the Fokker-Planck equation. This section describes two simple yet powerful variants, namely the Gaussian flow and Gromov flow. Finally, the advantages and disadvantages of the particle flow filter are summarized along with its algorithmic steps.

Section 5 describes the innovative PFF image registration algorithm proposed in this thesis project. The innovative aspect of this project is that image registration is viewed as a

Bayesian filtering problem and solved with the particle flow filter. There are several advantages of this method that are discussed in this section. The primary advantage is a computationally efficient method which reduces image registration misalignment caused by noise, outliers, and local minima. The authors of [1] proposed a similar method for image registration; however, [1] used the PF instead of the PFF. Unlike the method proposed in [1], particle resampling and importance weight updates are not required. Rather, the proposal density is formed by flowing the particles from the *a priori* to the *a posteriori*. The Gromov particle flow filter variant is used in this project for both 2D and 3D image registration. Although applicable to affine and elastic registration, this project focuses on rigid body image registration. This section concludes with the processing steps of the PFF image registration method.

Section 6 describes the implementation and results of the innovative PFF image registration method. The algorithms were implemented in MATLAB for both 2D and 3D rigid body registration with the Gromov particle flow filter variant. Additionally, the PF method proposed by [1] and ICP algorithms were implemented for comparison. All three registration techniques underwent Monte Carlo testing with high degrees of initial misalignment and noise. For the same alignment accuracy, the new particle flow filter algorithms were 244% faster than the particle filter for certain challenging problems. For the same alignment time, the particle flow filter reduced misalignment by as much as 35% compared to the particle filter. The particle flow filter achieved 100% alignment with enough particles, and reduced misalignment by as much as 75% over that of iterative closest point. These results demonstrate that image registration via the particle flow filter significantly outperforms the particle filter and iterative closest point algorithms in the presence of noise and a high degree of initial misalignment.

2 Overview of Image Registration

2.1 Overview

Image registration is an important function for many image processing applications. Image registration is the process of overlaying images of the same scene taken at different times, from different viewpoints, and/or by different sensors [4]. Image registration aims to find the optimal transformation to align 2D or 3D images by maximizing a measure of similarity. Alignment through image registration provides additional information and is needed for a variety of applications such as image fusion, change detection, and multichannel image restoration [4]. Additionally, image registration is a central task for applications such as medical image analysis, biomedical systems, stereo computer vision, optical flow estimation, autonomous driving, facial recognition, artificial intelligence, machine learning, target identification, cartography, remote sensing, and synthetic aperture radar [3]. Figure 1 taken from [5] shows the benefit of image registration. In this example, 1,004 photographs of a hazy Zurich skyline were registered to increase the signal to noise ratio (SNR) and reveal the Alps in the background.

Figure 1: Image Before (left) and After (right) Registration Reveal Alps in the Background [5]

There are a variety of registration methods available depending on the situation. These include feature-based registration and intensity-based registration for both rigid-body and deformable images. Feature-based registration aligns two images based on corresponding points, lines, contours, or other structures. Intensity-based registration aligns two images based on intensity information. Each registration method has its advantages and disadvantages and each method is application dependent; therefore, a universally optimal approach is impossible [4]. In the simplest terms, image registration is classified by dimensionality. 2D registration aligns pixels (i.e. picture elements) while 3D registration aligns voxels (i.e. volume elements).

A critical step in image registration is the coordinate transformation that aligns the two images. Typically, one image is held stationary and is referred to as the static or reference image. The other image referred to as the moving image is transformed to align it with the static image. The reference image has point set $A = \{ A_i \} = \{ A_1, A_2, \dots A_m \}$ and centroid $\mu_A = \frac{1}{m}\sum_{i=1}^{m} A_i$. The moving image has point set $B = \{ B_i \} = \{ B_1, B_2, \dots B_n \}$ and centroid $\mu_B = \frac{1}{n}\sum_{i=1}^{n} B_i$. As described in [6], image transformations are classified as follows:

1. **Rigid**: This registration type is linear and is described by translations and rotations, namely 2 rotations and 2 translations (2D) or 3 rotations and 3 translations (3D).

2. **Affine**: The affine transform includes translation, rotation, scaling, and shear. An affine transform is described by 8 parameters (2D) or 12 parameters (3D).

3. **Elastic**: Non-rigid and non-affine registration is called elastic. Elastic registration is also called curved, non-rigid, or deformable. This registration type is non-linear.

In addition to dimensionality and transformation type, image registration is classified by modality. Unimodal registration registers images of the same type, such as two computerized

axial tomography (CAT) scans. Multimodal registration refers to images of different types, such as registering a positron emission tomography (PET) scan onto a CAT scan [4].

Registration is typically a two-part problem: (1) determine the correspondence between two data sets, and (2) estimate the transformation parameters [1]. The correspondence represents similarity between images. There are three common measures of similarity between images: sum of squared distances between points, correlation between intensities, and mutual information [7]. The sum of squared distances is the simplest to calculate; however, it does not account for image intensity information or work well with multimodal registration [7]. Correlation similarity is efficient and utilizes image intensity information; however, it does not work well for multimodal registration [7]. Mutual information works better with multimodal images; however, it is computationally more intensive. Mutual information registration maximizes the shared information (i.e. entropy) between images [7].

The goal of image registration is to align two images in such a way as to optimize the desired measure of similarity. There are many methods described in literature for solving this optimization problem, but most are primarily based on the minimization of a cost function [3]. There are several optimization methods for localizing a minimum, such as linearizing the cost function or using multiscale spaces [3]. Much research has been conducted over the past decade to find new application specific registration methods, since registration establishes the baseline accuracy seen in subsequent image processing steps. Image registration methods vary based on anticipated geometric deformations, radiometric deformations, noise corruption, required registration accuracy, and data characteristics [4]. The next subsection of this report describes one popular method of image registration, namely the Iterative Closest Point algorithm.

2.2 Iterative Closest Point

An important algorithm for feature-based rigid-body registration is the Iterative Closest Point (ICP) algorithm. ICP works well for a variety of feature-based structures (i.e. points, lines, and contours); however, a common application is point sets (i.e. point-based registration, which is a subset of feature-based registration). Point-based registration is defined as the optimum transformation to align two data sets, given a set of matching points in two spaces [2]. ICP is discussed here in detail, because it is used within the PFF framework proposed in this paper.

The ICP algorithm is a popular technique and benchmark for point set registration [8]. It is a simple and robust algorithm that has many applications in rigid-body registration. As indicated by its name, ICP is an iterative approach for image registration. The objective of ICP is to minimize the Euclidean distance (i.e. L2) between features of two data sets. Given an initial alignment, ICP determines the correspondence, computes the transformation parameters, and proceeds in an iterative manner with the updated set of correspondences [8]. The algorithmic steps are described below and are summarized in the flow chart of Figure 2:

1. For each feature point in image B, find the closest point in image A.
2. Estimate the transformation which minimizes the mean-squared error of the distances found in the previous step using singular value decomposition (SVD).
3. Transform image B with the transformation found in the previous step.
4. Iterate until the algorithm converges on the optimal transformation that minimizes the square of distances between points in image A and image B.

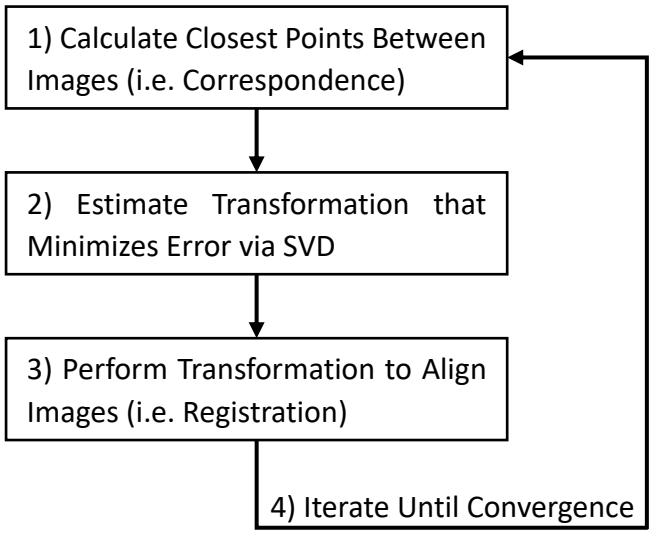

Figure 2: Iterative Closest Point Algorithmic Steps

There are ICP variants to efficiently determine the correspondence in step #1, such as utilizing a K-dimensional (K-D) tree which efficiently partitions data points. Additionally, ICP variants include methods for subsampling, weighting points, rejecting outliers, and using principle component analysis (PCA) for initial alignment. SVD is used in step #2 to calculate the transformation that aligns the correspondences given in step #1. This alignment is an eigenvalue problem, thus, SVD produces the correct transformation. The objective of the problem is to find the rotation matrix R and translation vector t that minimizes the error:

$$E = \sum_{i=1}^{n} \left|\left|A_i - B_i'\right|\right|^2 = \sum_{i=1}^{n} \left|\left|X_i - Y_i'\right|\right|^2 = \sum_{i=1}^{n} \left|\left|A_i - R\,(B_i) - t\right|\right|^2$$

where X and Y are the data after subtracting the centroids, so $X = A - \mu_A$ and $Y = B - \mu_B$. Note: $X = \{ X_i \}$ is arranged and truncated to correspond with the closest points of $Y = \{ Y_i \}$. The new image is B_i' so that $B_i' = R\,(B_i) + t$ and $Y_i' = R\,(Y_i)$. Solving for the objective function E leads to $A_i - B_i' = X_i - Y_i'$ which becomes $A_i - R\,(B_i) - t = (A_i - \mu_A) - R\,(B_i - \mu_B)$. After a little algebra, the translation vector is solved as $t = \mu_A - R\,(\mu_B)$.

In order to find the rotation matrix R, the cross-covariance matrix M is calculated:

$$M = X\,Y^T = \sum_{i=1}^{n} X_i Y_i^T$$

The error is minimized with the orthonormal matrix R that maximizes the trace $\mathrm{Tr}[R\,M]$. The SVD of $M = U\,W\,V^T$. The trace $\mathrm{Tr}[R\,M]$ is maximized with the rotation matrix $R = V\,U^T$.

Assuming the images are not perfectly aligned, there will be a residual error. Step #4 iterates until the root mean squared distance (RMSD) is minimized or until an error threshold is achieved. The RMSD error is:

$$E_{RMSD} = \sqrt{\frac{\sum_{i=1}^{n} \left|\left|A_i - B_i'\right|\right|^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} \left|\left|X_i - Y_i'\right|\right|^2}{n}}$$

Given the singular values σ_1, σ_2, and σ_3 of $W = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}$, the minimum error is:

$$E_{min} = \sum_{i=1}^{n} (||X_i||^2 + ||Y_i||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

The above discussion shows the mathematical development of the ICP algorithm. The reader is referred to [9] for a complete derivation with proof. As outlined above, the detailed steps for the ICP algorithm are to: center align the data sets to form X and Y by subtracting the centroids; arrange and truncate $X = \{ X_i \}$ to correspond with the closest points of $Y = \{ Y_i \}$; determine the cross-covariance matrix M, which equals $M = X\,Y^T$; calculate the SVD of M, which becomes $M = U\,W\,V^T$; form the rotation matrix R and translation vector t with the equations $R = V\,U^T$ and $t = \mu_A - R\,(\mu_B)$; transform the moving image as $B_i' = R\,(B_i) + t$; calculate the RMSD error; and repeat until the error is minimized or an error threshold is reached.

3 Particle Filter

3.1 Overview

The particle filter (PF) is a sequential Monte Carlo (MC) method that uses particles (point masses) and weights (importance) to represent the probability densities for Bayesian filtering. The PF was introduced in 1993 as a numerical approximation to the nonlinear Bayesian filtering problem [10]. PF theory is currently mature with many successful applications described in literature [10]. Like any Bayesian filtering scheme, the goal is to update the *a priori* prediction (i.e. *prior*) given a new measurement. The updated prediction is called the *a posteriori* estimate (i.e. *posterior*), which is the estimate that maximizes the *a posteriori* probability.

The vector x represents the target state, and the vector z represents the measurement. The probability density function (PDF) that represents the prior density is called the *prior*, $p_x(x)$. The conditional probability of the measurement given the state is called the *likelihood*, $p_z(z|x)$. The conditional probability of the state given the measurement is called the *posterior* or *conditional*, $p_x(x|z)$. The normalizing factor guaranteeing the total probability of $p_x(x|z)$ equals one is $p_z(z)$, where $p_z(z) = \int p_z(z|x)p_x(x)dx$. Therefore, the Bayesian framework becomes:

$$p_x(x|z) = \frac{p_z(z|x)p_x(x)}{p_z(z)}$$

Using the Chapman-Kolmogorov theorem, the equation relating current time to previous time is:

$$p_x(x_k|Z_k) = \frac{p_z(z_k|x_k)p_x(x_k|Z_{k-1})}{p_z(z_k|Z_{k-1})}$$

where the x_k and z_k vectors represent the state and measurement at time k, and the Z_k and Z_{k-1}matrixes represent all measurements of z up to and including k and k-1, respectively [11].

For particle filtering, the main goal is to represent the *a priori* and *a posteriori* densities by a set of weighted random samples and compute estimates based on these samples and weights [12]. As the number of samples approaches infinity, the PF solution becomes equivalent to the actual *a posteriori* PDF, and the PF achieves the optimal Bayesian estimate [12]. In practice, a sufficiently large number of particles approximates the actual PDF very closely and provides a near equivalent approximation. The PF uses recursive prediction and correction updates based on the *a priori*, *a posteriori*, and *likelihood* densities. In this sense, it is of the same class and has many similarities to the Kalman filter, the most famous of all Bayesian filters.

3.2 Kalman Filter

The author assumes the reader has a basic understanding of the Kalman filter; therefore, a summary of the Kalman filter is provided to aid in describing the particle filter. The reader is referred to [13] and [14] for a thorough overview of the Kalman filter. The Kalman filter is a set of equations that provides computationally efficient estimates to the state process by minimizing the mean squared error [14]. Even with unknown system models and measurement uncertainty, the Kalman filter is a powerful predictor of past, present, and future state estimates [14]. The recursive nature makes the Kalman filter appealing, because it does not need to maintain all data to make the optimal prediction [14].

Figure 3 adapted from [14] shows the algorithmic steps of the Kalman filter. There are two primary stages: "Predict" and "Correct", which correspond to the *a priori* and *a posteriori* estimates, respectively. The "hat" above the variable x designates it as an estimate (e.g. $\hat{x}$), and the superscript '-' designates an *a priori* estimate prior to the measurement update (e.g. P^-_k).

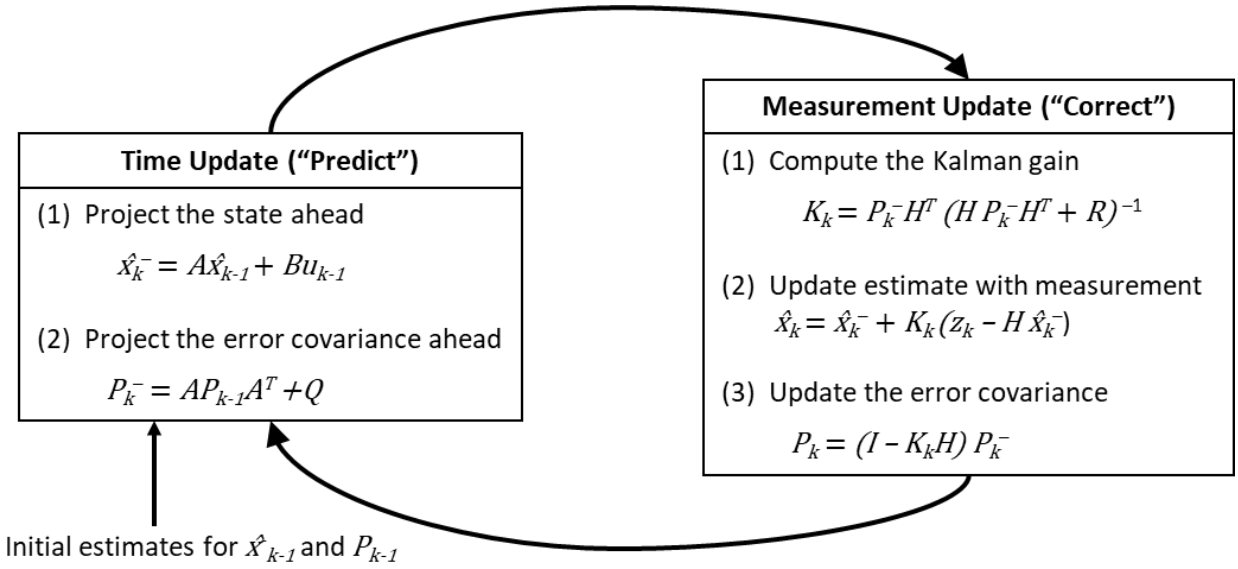

Figure 3: Kalman Filter Algorithmic Steps [14]

Like the particle filter, x and z are the state and measurement vectors, respectively. The state transition matrix A describes the evolution of the state with time. The measurement matrix H relates the measurements to the state. The error covariance matrix P is the estimate of the uncertainty in the Kalman prediction. The process noise covariance matrix Q accounts for uncertainty in the state model. The measurement error covariance matrix R accounts for the uncertainty in the measurements. The Kalman gain K is an adaptive parameter that either favors the *a priori* estimate or measurement based on the predicted error P and measurement error R.

The particle filter has its own representation for these parameters. Unlike the Kalman filter's closed-form equations, the particle filter uses particles to represent the densities. The Kalman filter assumes that the underlying statistics of the process noise and measurement error are unimodal, linear, and Gaussian. Additionally, the Kalman filter assumes that the *a priori* and *a posteriori* densities are Gaussian at every step. The Kalman filter provides the optimal solution to linear prediction, assuming these highly restrictive assumptions hold [12]. This means that no algorithm can beat the Kalman filter in the unimodal, linear, and Gaussian environment [12].

These restrictive assumptions do not hold in many real-world scenarios, and the Kalman filter must make approximations [12]. There are several Kalman filter variants that make nonlinear approximations, such as the extended Kalman filter (EKF). The EKF has been referred to as the "workhorse" algorithm for nonlinear filtering problems [15]. The EKF approximates the non-linearity with a local linearization using the first term in a Taylor expansion [12]. One fundamental flaw of the EKF is that the distributions are no longer normal after undergoing the respective nonlinear transformations [14]. In this sense, the EKF is a suboptimal state estimator that only approximates Bayes' rule by linearization [14]. For this reason, the EKF can deliver poor performance in many real-world situations.

3.3 Particle Filter

The particle filter is an effective method that does not assume unimodal, linear, or Gaussian densities. For two decades, the PF has been the most popular approach for nonlinear and non-Gaussian state estimation [16]. Increasingly, many applications need to more accurately model the underlying dynamics of a system with nonlinear and non-Gaussian distributions [12]. In this sense, the particle filter is a generalization of the Kalman filter and is not limited to the same restrictive assumptions. As the number of samples increases, the approximated *a posteriori* density approaches the true *a posteriori* density [12].

The particle filter begins by initializing N_s particles. These particles are typically drawn from a uniform distribution over the possible range of values for that state variable. If prior information is known about the state, then particles are initialized in a more educated fashion. Each particle is assigned a weight, which upon initialization is typically $1/N_s$. At each iteration,

the particles are propagated forward in time via the propagation function (i.e. motion model or state equation) f, where $x_k = f(x_{k-1})$. This propagation function corresponds to the Kalman filter's state transition matrix A. The collection of particles and weights form the *a priori* density, $p_x(x_k|Z_{k-1})$. The particles are then transformed into the measurement space via the measurement function, H, where $z_k = H(x_k)$. Besides going state-to-measurement, H is equivalent for the Kalman filter's measurement function H which goes measurement-to-state.

Next, the likelihood function $p_z(z_k|x_k)$ is calculated given a new measurement. This is done statistically based on the measurement covariance R and error between the measurement and particle. The particles closer to the new measurement have a higher probability than those further away. Each individual likelihood is the probability of the error given the measurement covariance (i.e. variances). Next, the particle weights are multiplied by the likelihoods and renormalized by the total probability, $p_z(z_k|Z_{k-1})$. The updated particles comprise the *a posteriori* density, $p_x(x_k|Z_k)$. That concludes the Bayesian filtering process:

$$p_x(x_k|Z_k) = \frac{p_z(z_k|x_k)p_x(x_k|Z_{k-1})}{p_z(z_k|Z_{k-1})}$$

The *a posteriori* state estimate $\hat{x}_k$ can now be calculated. There are several methods to estimate the state, but one common method is to take a weighted average of the particles. Additionally, the error covariance P is easily calculated by finding the variance of particles. The error covariance is equivalent for the Kalman filter. This process continues for each new measurement.

There are two inherent problems with the particle filter, namely particle degeneracy and particle impoverishment. Particle degeneracy occurs when the variance of the importance weights increases with time and the weights approach zero. This leads to little or no coverage over the area of required density. Particle degeneracy is unavoidable because the variance of

weights only increases with time [12]. Additionally, particle degeneracy worsens with more accurate measurements [15]. The brute force solution to particle degeneracy is to increase the number of particles. However, with the increase of particles, the computational burden grows exponentially because of the curse of dimensionality [15]. Figure 4 taken from [15] illustrates particle degeneracy. As can be seen, there are no *a priori* particles (green dots) in the region needed for the product of the *a priori* density and *likelihood* function (blue and red curves) [15].

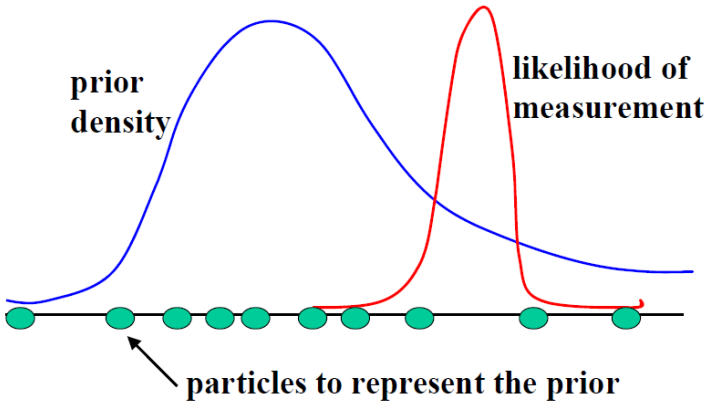

Figure 4: Particle Degeneracy Illustrated [15]

The methods typically deployed to combat particle degeneracy are sequential importance sampling (SIS) and resampling [16]. For SIS, the particles more accurately representing the likelihood are given higher weights. The SIS algorithm consists of propagating the weights forward in time, receiving a new measurement, assigning a probability of that particle given the new measurement, and updating the particle weight by multiplying the previous weight with the new probability. For resampling, particles with low probability (i.e. small weights) are discarded and resampled to higher probability (i.e. larger weights). Resampling eliminates particles that have small weights in order to concentrate more particles around areas of higher probability.

Another inherit problem of particle filtering is particle impoverishment, which is a consequence of resampling. Resampling reduces the degeneracy problem; however, it

introduces the problem of particle impoverishment [12]. Particle impoverishment leads to the duplication of particles and a loss of particle diversity [12]. With small process noise, all particles collapse to a single point within several iterations [12]. There are techniques used to solve particle impoverishment, such as the resample-move algorithm, bridging densities, and progressive correction [12]. The method used in this project for the PF approach is regularization, which adds a small value (i.e. dither) taken from a Gaussian draw onto the resampled particles. This is analogous to the Kalman Filter's process noise covariance matrix Q. Regularization increases diversity of the resampled points and reduces particle impoverishment.

As discussed, the particle filter is an excellent choice for many unimodal, nonlinear, and non-Gaussian filtering problems. However, the particle filter comes with inherit challenges, namely particle degeneracy, particle impoverishment, and the curse of dimensionality. For high dimensional problems, the accuracy can be orders of magnitude worse than optimal [15]. Like the Kalman filter, the particle filter is a Bayesian filter, but with less restrictive assumptions.

The following steps summarize the traditional particle filter algorithm:

(1) Project the particles forward in time via the propagation function, f, where $x_k = f(x_{k-1})$

(2) Determine the likelihood $p_z(z_k|x_k)$ of each particle given a new measurement z_k

(3) Update the particle weights and renormalize to form the proposal density, $p_x(x_k|Z_k)$

$$p_x(x_k|Z_k) = \frac{p_z(z_k|x_k)p_x(x_k|Z_{k-1})}{p_z(z_k|Z_{k-1})}$$

(4) Calculate the state estimate $\hat{x}_k$ based on the particles (e.g. weighted average)

(5) Resample: randomly select particles according to their weights and weight all as $1/N_S$

(6) Regularize: add a small value (i.e. dither) to the resampled particles from a Gaussian draw

Figure 5: Particle Filter Algorithmic Steps

4 Particle Flow Filter

4.1 Overview

The particle flow filter (PFF) was designed to avoid the shortfalls of the particle filter. This new class of nonlinear filters has gained attention in the past several years and was invented by Fred Daum and Jim Huang in order to solve the problem of particle degeneracy with the PF [16]. The PFF flows the particles from the *a priori* to the *a posteriori* via partial differential equations (PDEs), and it achieves the advantages of the PF without the consequences of degeneracy [15]. Particle weight updates, resampling, and regularization are not required. For certain problems, the PFF has been shown to be faster and more accurate [18]. The PFF begins with Bayes' Law:

$$p_x(x|z) = \frac{p_z(z|x)p_x(x)}{p_z(z)}$$

In order to simplify the notation and follow the convention outlined in [17] and [19], the following abbreviations are made: $p(x) = p_x(x|z)$, $m(x) = p_z(z|x)$, $g(x) = p_x(x)$, and $K = p_z(z)$.

$$p(x) = \frac{m(x)g(x)}{K}$$

If the distributions $m(x)$ and $g(x)$ are nowhere vanishing over a common region, the natural logarithm of Bayes' rule provides an efficient alternative, thus preventing singularities [19]:

$$\log p(x) = \log g(x) + \log m(x) - \log K$$

Next, the parameter λ is introduced to define the homotopy (i.e. flow) of densities for $0 \leq \lambda \leq 1$.

$$\log p(x,\lambda) = \log g(x) + \lambda \log m(x) - \log K(\lambda)$$

In order to simulate time, the scalar valued parameter, λ, is introduced into the homotopy equation [19]. The parameter λ plays the role of time and varies from 0 to 1 [19]. During filtering, Bayes' rule operates over actual time each time a measurement is received. At each time

increment, a synthetic loop of time is performed to move the *a prior density* to the *a posteriori*. Because this loop is not actual time in relation to the system dynamics, λ is considered a simulated or synthetic time variable. During the homotopy process, λ is divided into smaller steps. Several methods exist to choose the step size $d\lambda$. The simplest is to take equal steps. Another method is to increase the step size exponentially over the interval 0 to 1 [18].

The normalization constant K is parameterized by λ so that $p(x, \lambda)$ is a valid PDF for all values of λ [17]. The normalizing factor K is not necessary for the PFF as the goal is to maximize the function; therefore, the normalized conditional probability is not actually computed. Instead, the log of the unnormalized conditional density is calculated, which avoids numerical problems [19]. The unnormalized homotopic logarithmic Bayes' equation is as follows:

$$\log p(x, \lambda) = \log g(x) + \lambda \log m(x)$$

When λ = 0, then the equation becomes $\log p(x, 0) = \log g(x)$, which is the unnormalized *a priori* density. When λ = 1, then the equation becomes $\log p(x, 1) = \log g(x) + \log m(x)$, which is the unnormalized *a posteriori* density. For λ between 0 and 1, the homotopic flow occurs.

The key to the PFF is to accurately model the flow of particles. The PFF assumes that the particles flow according to the following Ito stochastic differential equation [17]:

$$dx = f(x, \lambda)d\lambda + B(x, \lambda)dw_\lambda$$

where f is the drift function and B is the diffusion matrix function. If $B(x, \lambda)$ is zero, then the equation corresponds to the deterministic differential equation $dx/d\lambda = f(x, \lambda)$; however, if $B(x, \lambda)$ is nonzero, then the equation is stochastic and x follows a random path [17].

The goal of the PFF designer is to choose the functions $f(x, \lambda)$ and $B(x, \lambda)$ that have a corresponding Fokker-Planck equation such that integrating λ from 0 to 1 results in $p(x, 1)$ being

identical to the Bayesian solution [17]. The Fokker-Planck equation is a partial differential equation with drift and diffusion functions that model the evolution of a PDF [17]. Using the stated functions, the N variable Fokker-Planck equation is [17] [20]:

$$\frac{\partial p(x,\lambda)}{\partial \lambda} = -\sum_{i=1}^{N} \frac{\partial(\, f(x,\lambda) p(x,\lambda)\,)}{\partial x_i} + \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial^2 \big(\, B(x,\lambda) p(x,\lambda)\big)}{\partial x_i \partial x_j}$$

The following relationship exists between the diffusion matrix and process noise covariance [17]:

$$B(x,\lambda) = \frac{1}{2}\sum_{k=1}^{N} \sigma_{ik}\sigma_{jk} = \frac{1}{2}\sum_{k=1}^{N} Q$$

In vector form, the Fokker-Planck equation then becomes [21]:

$$\frac{\partial p(x,\lambda)}{\partial \lambda} = -\mathrm{div}\big(f(x,\lambda)\; p(x,\lambda)\big) + \frac{1}{2}\mathrm{Tr}\left[Q\,\frac{\partial^2 p(x,\lambda)}{\partial x^2}\right]$$

$$\frac{\partial p(x,\lambda)}{\partial \lambda} = -\mathrm{div}\big(f(x,\lambda)\; p(x,\lambda)\big) + \frac{1}{2}\mathrm{div}(Q\,\frac{\partial p(x,\lambda)}{\partial x})$$

where $\mathrm{div}(\cdot)$ is the divergence. There are many methods to solve the Fokker-Planck equation and each method provides a different PFF variant [19].

4.2 Variants

The choice of flow can drastically affect the functionality of the algorithm [17]. The goal of the PFF designer is to find a solution that is simple and numerically stable when integrated [17]. PFF variants arise from the method and assumptions used to solve the Fokker-Planck equation. It is typically difficult to obtain a solution directly, especially if separation of variables is not possible or if the number of variables is large [20]. For a thorough discussion of variants, the reader is referred to [17] [18] [21] [22] [23] [24]. This section describes two variants, namely the Gaussian flow and Gromov flow.

The Gaussian flow PFF is a zero diffusion PFF variant with a simple explicit solution. This variant is often called the exact flow in several papers [17] [18] [24]. If the diffusion term $B(x, \lambda)$ is assumed zero, then the Fokker-Planck equation simplifies considerably. The Gaussian flow assumes that the *a priori*, *a posteriori*, and *likelihood* distributions are Gaussian [17]. The complete derivation is provided in [24]. The Gaussian flow is stated below [17]:

$$f = Ax + b$$

$$A = -\frac{1}{2}PH^T(\lambda HPH^T + R)^{-1}H$$

$$b = (I + 2\lambda A)(A\hat{x}_{pred} - (PH^T R^{-1} + \lambda APH^T R^{-1})v)$$

$$v = h(x) - z$$

Although the Gaussian flow PFF is simple, it is nonideal for certain applications. In addition to the restrictive assumptions of zero diffusion and Gaussian densities, the Gaussian flow has been shown to be inherently biased [17]. Nonetheless, this PFF works well for many applications.

The Gromov flow PFF is a nonzero diffusion PFF variant. This PFF variant is derived using Gromov's Theorem, which solves linear underdetermined smooth PDEs without integration if and only if the PDE is smooth and the number of unknowns is sufficiently large (i.e. at least the number of linearly independent equations plus the dimension of the state vector) [18]. The Gromov flow PFF makes no assumptions on the PDFs and is valid for any smooth nowhere vanishing densities [18]. The Gromov flow PFF variant is derived in [18] and defined below [18]:

$$f = -\left[\frac{\partial^2 \log p}{\partial x^2}\right]^{-1}\left[\frac{\partial \log h}{\partial x}\right]^T$$

$$Q = -\left[\frac{\partial^2 \log p}{\partial x^2}\right]^{-1}\left[\frac{\partial^2 \log h}{\partial x^2}\right]\left[\frac{\partial^2 \log p}{\partial x^2}\right]^{-1}$$

If the *a priori* density is assumed Gaussian, then the Gromov flow PFF simplifies into a simpler explicit solution. This simplification proves highly accurate for many nonlinear, non-Gaussian, and multimodal densities; however, the final choice of PFF variant is scenario specific. The simplified Gromov flow variant is derived in [18] [24], and stated below [17]:

$$f = -(P^{-1} + \lambda H^T R^{-1} H)^{-1} H^T R^{-1} (h - z)$$

$$Q = (P^{-1} + \lambda H^T R^{-1} H)^{-1} H^T R^{-1} H (P^{-1} + \lambda H^T R^{-1} H)^{-1}$$

The PFF variants are solutions to the Fokker-Planck equation. f and Q model the flow of particles, and must be integrated over λ from 0 to 1 to obtain the updated particles. There are several methods to select the step size $d\lambda$ and integrate the flow of densities. The reader is referred to [17] for a discussion of these techniques. This project uses equal and logarithmic step sizes, and the integration method used is the explicit strong Euler-Maruyama method [17]:

$$x_{\lambda + d\lambda} = x_\lambda + f(x_\lambda, \lambda) d\lambda + B(x_\lambda, \lambda) \tilde{w}$$

where $\tilde{w}$ is a zero-mean Gaussian random variable with covariance matrix $d\lambda \cdot I$. The Gaussian flow and Gromov flow explicit functions for f and B are plugged into the Euler-Maruyama equation at each step $d\lambda$. For the Gaussian flow, the diffusion term B is zero. For the Gromov flow, B is obtained from the equation $Q = BB^T$. B is essentially the matrix square root of Q, which can be solved via Cholesky decomposition, LDL decomposition, or other methods [17].

4.3 Advantages & Disadvantages

The advantage of the PFF over other filtering algorithms is the computational speed and accuracy for high dimensional nonlinear applications. The PFF has been shown to be many orders of magnitude faster than the standard PF for high dimensional problems [16] [18] [25].

Additionally, the PFF significantly outperforms the EKF in accuracy for certain nonlinear problems [16] [18]. The primary advantage of the PFF is its higher computational speed and higher accuracy for nonlinear, multimodal, non-Gaussian, and high dimensional problems.

The reason for the higher computation speed is threefold. First, the PFF requires far less particles to achieve the same accuracy as the PF [15]. Second, resampling and weight updates are not required. Third, PFF processing is parallelizable unlike the PF where resampling creates a bottleneck [15] [16]. This allows for the effective utilization of highly efficient graphics processing units (GPUs). The PF bottleneck becomes more apparent due to the curse of dimensionality for the PF, which states that the computational burden increases exponentially as the dimensionality increases. The PFF does not suffer from this curse.

There are several downsides to the PFF algorithm. Although particle degeneracy is impossible, particles can move to regions that poorly represent the *a posteriori* density [17]. Additionally, certain PFF variants have been shown to be inherently biased [17] [24] or suboptimal [26]. Most of these problems are resolved by proper selection of the PFF equations, Fokker-Planck assumptions, state model, measurement model, covariances, and number of particles. Therefore, meticulous PFF design is essential for optimal filtering. The PFF cannot be treated as a generic black box that works for all state models [24].

The following steps summarize the particle flow filter algorithm:

(1) Project the particles forward in time via the propagation function, f, where $x_k = f(x_{k-1})$

(2) Determine the likelihood $m(x)$ of each particle given a new measurement z_k

(3) Flow the *a priori* density $g(x)$ to the *a posteriori* $p(x)$ via the Fokker-Planck equation

(4) Calculate the state estimate $\hat{x}_k$ based on the particles (e.g. mean of particles)

Figure 6: Particle Flow Filter Algorithmic Steps

5 Registration via Particle Flow Filter

5.1 Overview

Image registration is ideally suited to fit within the Bayesian filtering framework. The iterative process of prediction and correction naturally turns image registration into a *posterior* estimation problem. Figure 7 shows ICP registration within the Bayesian process. Much research has been conducted over the past decade to improve image registration with several methods treating image registration as a *posterior* estimation problem [1] [2] [3]. However, no method uses the particle flow filter for image registration. The novelty of this project is that image registration is viewed as a Bayesian filtering problem and solved with the particle flow filter.

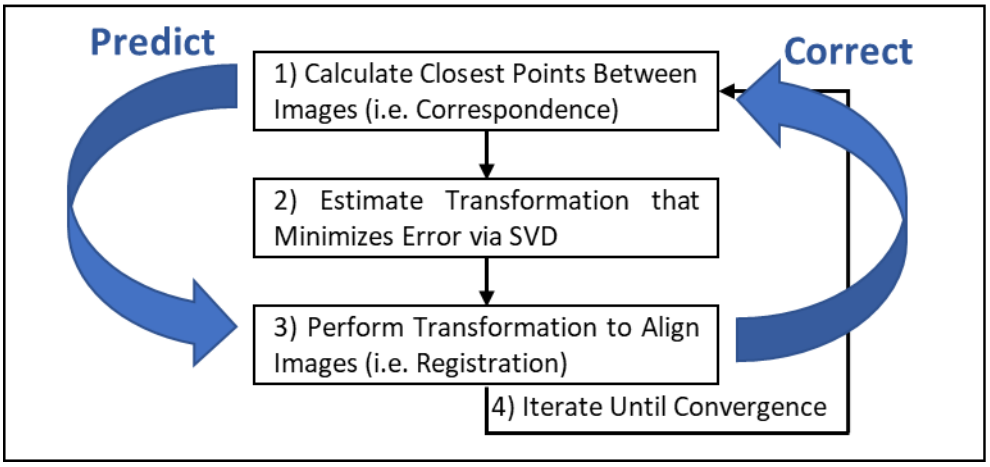

Figure 7: ICP Registration within Bayesian Framework

There are several advantages of point-set registration via the PFF. Like many registration algorithms, ICP registration is susceptible to converging on a local minima, which results in misalignment. One solution is to place highly restrictive initial alignment requirements on the images to limit misalignment, but this solution requires prior information about the image, scene, and geometry. Geometric assumptions and initial alignment are not required for the PFF

approach. Another solution is to calculate an error metric to reveal a local minima. This solution is nonideal because it assumes a certain similarity between images. Differences between the two images regarding pose, noise, or mode can result in a large error metric even though the solutions converge properly. Additionally, the error metric for a local minima might fall below the threshold, and go undetected. With enough particles, the PFF avoids this problem.

Another solution to prevent misalignment is to use a multiple hypothesis or Bayesian filtering approach that maintains multiple solutions in attempts to find the global minima. Unlikely hypotheses are removed based on low *a posteriori* densities. This is the method employed in this project. Other registration techniques use the Bayesian filtering approach; however, most assume unimodal, linear, and Gaussian densities. Unfortunately, these highly restrictive assumptions are often invalid. The PF does not make these assumptions and has been employed in image registration by several authors [1] [3]. However, the PF is slow due to resampling and weight updates. The PFF provides a faster and more accurate alternative.

5.2 PF Baseline

Of relevance to this project is the research conducted by Sandhu, Dambreville, and Tannenbaum from Georgia Institute of Technology [1] [8]. These authors developed a method for point-set rigid body registration guided by the particle filter. Due to the similarities between Sandhu *et al*. and the innovative PFF method outlined in this thesis project, a brief explanation of their method is provided. For a more thorough explanation, see [1] and [8].

The objective function proposed by Sandhu *et al*. is a local optimizer based on the correlation between two point-sets. This objective function is a robust ICP variant based on a

Gaussian mixture model where outliers are penalized by an exponential term [1]. The goal is to minimize the objective function. Like the ICP algorithm described in section 2.2, the robust ICP variant continues until convergence is reached. The objective function [1] is given by:

$$E = \lambda \sum_{i=1}^{N_d} \omega_i \exp\left(-\frac{1}{4}\left|\left|Rn_i^* \cdot [A_i - RB_i^* - t]\right|\right|^2\right)$$

where $\lambda = \frac{1}{(2\pi)^{N/2}2\sqrt{2}}$; ω_i is a quantity describing the Gaussian mixtures ($\omega_i = \beta_i \alpha_i^*$); R is the rotation matrix; t is the translation vector; B_i is the moving image; and A_i is the reference image.

The state space is the transformation matrix (i.e. rotation and translation). The motion model is the ICP algorithm. Each particle is propagated forward in time by *L* iterations of the ICP algorithm. Time, *t*, is an artificial parameter where "information" is the point correspondences for a given pose estimate as the registration algorithm converges [1]. The particle importance weights are updated based on an exponential form of the misalignment error, which keeps the algorithm robust to outliers. The exponential form of the misalignment error is stated below [8]:

$$p(z_t|\hat{x}_t) \triangleq \exp\left(\sum_{i=1}^{N_d}\left|\left|A_i - T(B_i)\right|\right|^2\right)$$

The measurement is the image formed by the original moving image and transformation matrix that minimizes the objective function (i.e. the particle with the highest importance weight) [8]. In the state space, the measurement is the transformation matrix [8]. The motion alignment error (*e*) for each particle is the error between the predicted ($\hat{x}$) and measured (*x*) state [1]. The motion error covariance (*S*) is the autocorrelation of the alignment error, where *S* describes the uncertainty in each of the principle axis of the rigid body transformation [1]. The motion alignment error, *e*, and motion error covariance, *S*, are as follows [1]:

$$e(x^i_{t-1}, \hat{x}^i_{t-1}) = x^i_{t-1} - \hat{x}^i_{t-1}$$

$$S^i_{t-1} = \mathrm{E}[e(x^i_{t-1}, \hat{x}^i_{t-1})\, e(x^i_{t-1}, \hat{x}^i_{t-1})^T]$$

Sandhu *et al*. assume that the diffusion term (i.e. process noise) is initially nonzero (i.e. the *a posteriori* density follows stochastic motion); however, as *t* approaches infinity, it reduces to zero (i.e. deterministic motion), which allows for convergence as the process noise decreases ($\sigma^2_{t-1} \rightarrow 0$) [1]. The *a posteriori* density (i.e. proposal density) is as follows [1]:

$$q(x_t | x^i{}_{t-1}, z_k) = \frac{1}{N}\sum_{i=1}^{N} K\left(\frac{x^i{}_t - x^i{}_{t-1}}{\gamma_{t-1} \cdot S^i{}_{t-1}}\right)$$

$$K(x^i{}_t, x^i{}_{t-1}, h^i{}_{t-1}) = \frac{\exp\left(\frac{1}{2}(x^i{}_t - x^i{}_{t-1})^T (h^i{}_{t-1})^{-1}(x^i{}_t - x^i{}_{t-1})\right)}{(2\pi)^{N/2}|h^i{}_{t-1}|^{\frac{1}{2}}}$$

$$h^i{}_{t-1} = \gamma_{t-1} S^i{}_{t-1} + v_{t-1}$$

where $h^i{}_{t-1}$is the bandwidth of the Gaussian Kernel K; $\gamma_{t-1}$is the diffusion weight which is typically assumed one [8]; and $v_{t-1}$is the process noise which is typically assumed zero [8].

Figure 8 summarizes the PF registration algorithm proposed by Sandhu *et al*. [8]:

For t = 0	**For t = 1, 2, 3, …**
(1) Initialize by drawing N particles	(1) Draw N particles from $q(x_t \| x^i{}_{t-1}, z_k)$
(2) Propagate particles through $z_0 = h(\hat{x}^i_0, C^i(0))$ – Perform L steps of $f_{ICP}(A, B, \hat{x}^i_0)$ – Update Weights via ${w_0}^i \propto p(z_0 \| \hat{x}^i_0)$ – Resample particles – Compute $z_0 = min\ \{ arg\{ f_{ICP}(A, B, \hat{x}^i_0)\}\}$	(2) Propagate particles through $z_t = h(\hat{x}^i_t, C^i(t))$ – Perform L steps of $f_{ICP}(A, B, \hat{x}^i_t)$ – Update Weights via ${w_t}^i \propto p(z_t \| \hat{x}^i_t)$ – Resample particles – Compute $z_t = min\ \{ arg\{ f_{ICP}(A, B, \hat{x}^i_t)\}\}$

Figure 8: Particle Filter Image Registration Algorithm [8]

5.3 PFF Method

As discussed in section 4, the PFF provides a faster alternative to the PF. The flow of particles from the *a priori* to the *a posteriori* prevents particle degeneracy and the associated computation burden of particle resampling and weight updates. This thesis project approaches image registration as a *posterior* estimation problem. The innovative aspect of this project is the use of the PFF to guide image registration. Although applicable to affine and elastic registration, this project focuses on 2D and 3D rigid body image registration.

This thesis project uses a similar framework to the PF method discussed in section 5.2, but with many modifications. The PFF method proves just as robust to noise, outliers, and local minima. The objective function used in this project is the sum of distances squared:

$$E = \sum_{i=1}^{n} \left| |X_i - Y_i'| \right|^2$$

This objective function is computationally more efficient than the method proposed in [1] and more efficient that the root mean squared distance (RMSD) discussed in section 2.2. The RMSD is minimized when the sum of distances squared is minimized, thus the square root and scalar division is unnecessary. The PFF method uses *L* iterations of the ICP algorithm as the motion model that propagates each particle forward in time. The ICP variant used in this project is the one discussed in section 2.2. Although this project uses ICP as the underlying registration algorithm, the PFF registration method is suitable for a variety of registration techniques. The particle that minimizes the objective function forms the best estimate for that iteration.

Unlike the method proposed in [1], the PFF does not required particle resampling and importance weight updates. Rather, the proposal density is formed by flowing the *a priori* to the

a posteriori. This is done via the particle flow filter as described in section 4. Although there are a variety of PFFs, this thesis project uses the Gromov flow variant discussed in section 4.2. Since the measurement and state contain the same transformation parameters (i.e. rotation and translation), the measurement matrix (H) is simply the identity matrix (I). The measurement noise covariance matrix (R) is formed from the best particle's motion error covariance (S). A small value (i.e. 1e-5) is added to the measurement noise covariance (R) when a main diagonal term becomes zero. This small value is needed very infrequently (it was only seen once during 3D registration testing); however, it prevents singularities during the R matrix inversion.

Following [8], the process noise is assumed zero. The error covariance (P) is the particle variance multiplied by the identity matrix (I). When the particle variance collapses to zero in any single dimension, a singularity originates in the error covariance matrix inversion. This singularity becomes problematic when solving for B in the Gromov flow equation $Q = BB^T$ (see section 4.2). To avoid problems associated with this singularity, Cholesky decomposition is used to solve for B. Finally, although the step size $d\lambda$ is a design parameter, the two step methods used in this project are equal steps and logarithmic steps. The number of steps was also varied for optimality.

Figure 9 summarizes the innovative PFF registration algorithm:

For t = 0
(1) Initialize by drawing N particles
(2) Propagate particles through $z_0 = h(\hat{x}^i_0, C^i(0))$ – Perform L steps of $f_{ICP}(A, B, \hat{x}^i_0)$ – Compute $z_0 = min\ \{ arg\{ f_{ICP}(A, B, \hat{x}^i_0)\}\}$

For t = 1, 2, 3, …
(1) Flow particles via the particle flow filter
(2) Propagate particles through $z_t = h(\hat{x}^i_t, C^i(t))$ – Perform L steps of $f_{ICP}(A, B, \hat{x}^i_t)$ – Compute $z_t = min\ \{ arg\{ f_{ICP}(A, B, \hat{x}^i_t)\}\}$

Figure 9: Particle Flow Filter Image Registration Algorithm

6 Implementation & Results

6.1 Implementation

The PFF image registration method was implemented using MATLAB. As a means of comparison, the PF method outlined in [1] was implemented using the code provided by [27] as a starting point. The PF code was modified to conform to the processing steps outlined in section 3.3. Once PF functionality was verified, the PFF tool was created by removing the resampling, regularization, and adding the simplified Gromov PFF variant as discussed in section 4.2. Figure 10 shows the block diagram of the PF and PFF algorithms implemented in MATLAB:

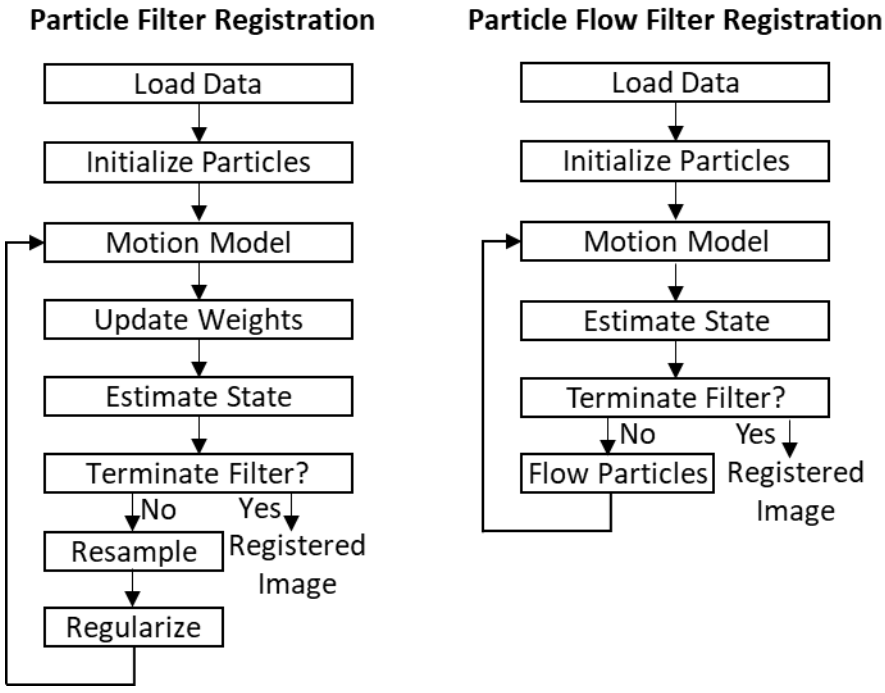

Figure 10: MATLAB Implementation of PF and PFF Registration

After the two MATLAB tools were developed, performance was compared between the PF and PFF methods. Additionally, performance was compared to the traditional ICP algorithm with no PF or PFF implementation. Initial misalignment and noise were added during Monte

Carlo testing. The images used in testing were taken from the Stanford University Graphics Archive [28], and included the 2D bunny, 2D brain, and 3D bunny. Figure 11 shows an example of the original Stanford Bunny [28] and a transformed version used as the moving image. The Stanford bunny was used for both 2D and 3D image registration. Figure 12 contains the Stanford brain [28] which is a cross section of the brain, skull, and surrounding anatomy. This image was used for 2D registration and contains more complex internal structures than that of the bunny.

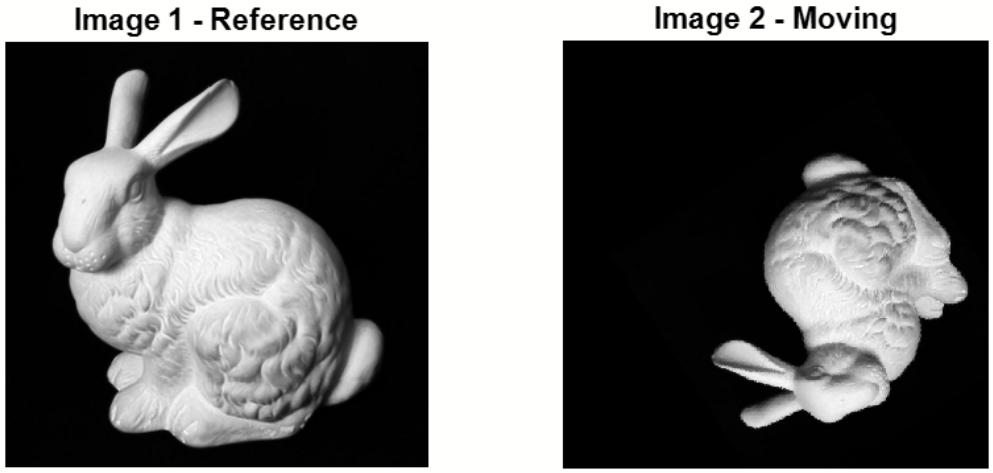

Figure 11: Original (left) and Initial Transformation (right) of the Stanford Bunny [28]

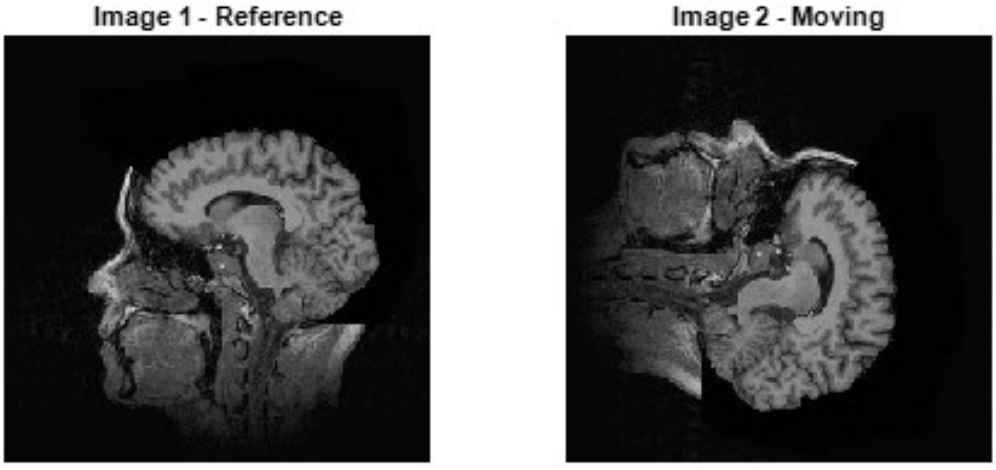

Figure 12: Original (left) and Initial Transformation (right) of the Stanford Brain [28]

Monte Carlo testing consisted of a variety of tests for all three of the images described above. Each Monte Carlo run consisted of 100 random trials, where the moving image was formed from a random transformation of the reference image. The random transformation introduced a large degree of initial misalignment. The initial transformation was composed of a translation and rotation. The translation came from a uniformly distributed random draw bounded by the dimensions of the reference image, $\sim U(A_{min}, A_{\max})$. The rotation came from a uniformly distributed random draw between 0 and 360°, $\sim U(0°, 360°)$. In addition to the initial misalignment, each trial added white gaussian noise (WGN) to each pixel location, $\sim N(B_i, \sigma^2)$. Three noise levels were used: no noise, medium noise, and high noise. The no noise experiments did not add noise to either image. The medium noise experiments added WGN to only the moving image. The high noise experiments added WGN to the reference and moving images.

Edge detection was conducted on the images prior to using them within the registration algorithms. This method was computationally more efficient and equivalently accurate as using the entire image due to the rigid body transformation. Below are examples of the three image types (i.e. 2D bunny, 2D brain, and 3D bunny) after edge detection with the three noise levels:

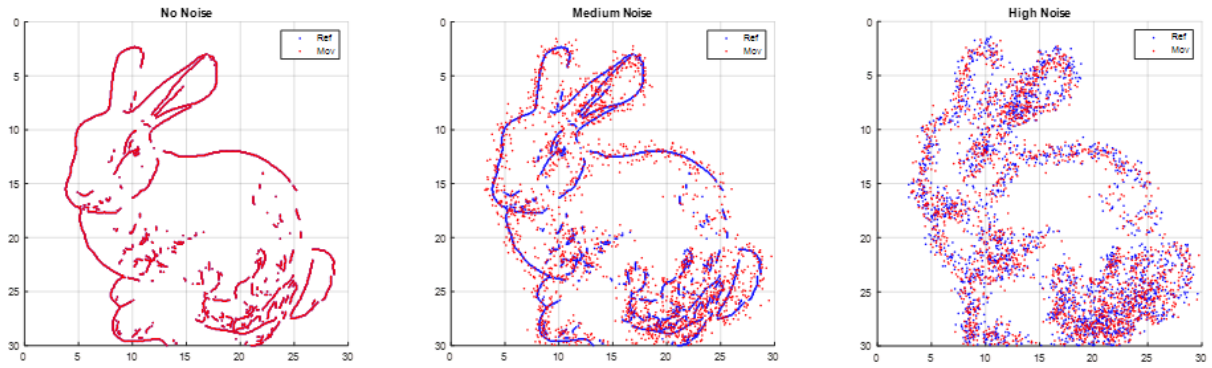

Figure 13: Successful Registration of the 2D Bunny with the Three Noise Levels

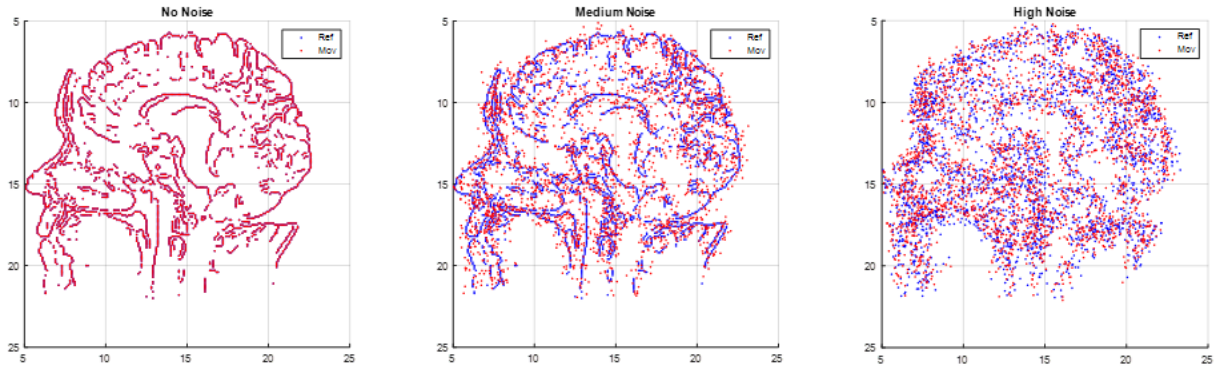

Figure 14: Successful Registration of the 2D Brain with the Three Noise Levels

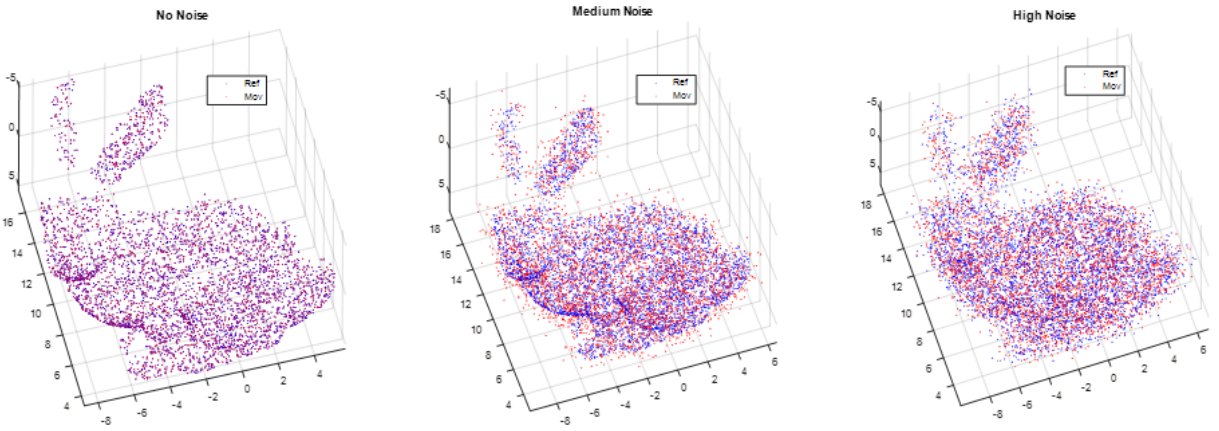

Figure 15: Successful Registration of the 3D Bunny with the Three Noise Levels

For each image type (i.e. 2D bunny, 2D brain, and 3D bunny), two tests were conducted. The primary set of tests varied the number of particles, and compared the PF, PFF, and ICP only registration. Besides the number of particles, all other parameters remained the same. The number of particles varied from 5 to 1,000. Initial misalignment and noise were added per the Monte Carlo method discussed above. The goal of this test was to compare speed and misalignment as a function of the number of particles for the PF, PFF, and ICP only registration. The secondary set of tests analyzed the PFF in more detail. This set of tests analyzed PFF performance as a function of number of λ steps and step method. The number of λ steps ranged

from 3 to 100,000 steps, and the two step methods were equal steps and logarithmic steps. The goal of this test was to compare speed and misalignment as a function of step size δλ and step method for the PFF. This test provided insight into the stiffness of the particle flow.

Testing was automated to determine if registration resulted in perfect alignment between images. The threshold used for declaring alignment and misalignment was based on the ICP minimum error, E_{min}, criteria discussed in section 2.2. The student verified that the thresholds accurately identified misalignment for all three image types and all three noise levels. Additionally, misalignment was declared if the images did not align after 30 iterations.

Figure 16 shows an example of PFF image registration implemented in MATLAB. The plot on the left shows the initial images, reference (blue) and moving (red). This example added no noise to the pixels, so the moving image is a randomly rotated and translated version of the reference. The particles associated with the plot on the left show the particles initialized uniformly. The plot on the right shows the final images perfectly aligned after 22 iterations, thus the reference image (blue) is completely covered by the moving image (red). The final particles are grouped near the proper transformation (i.e. solution).

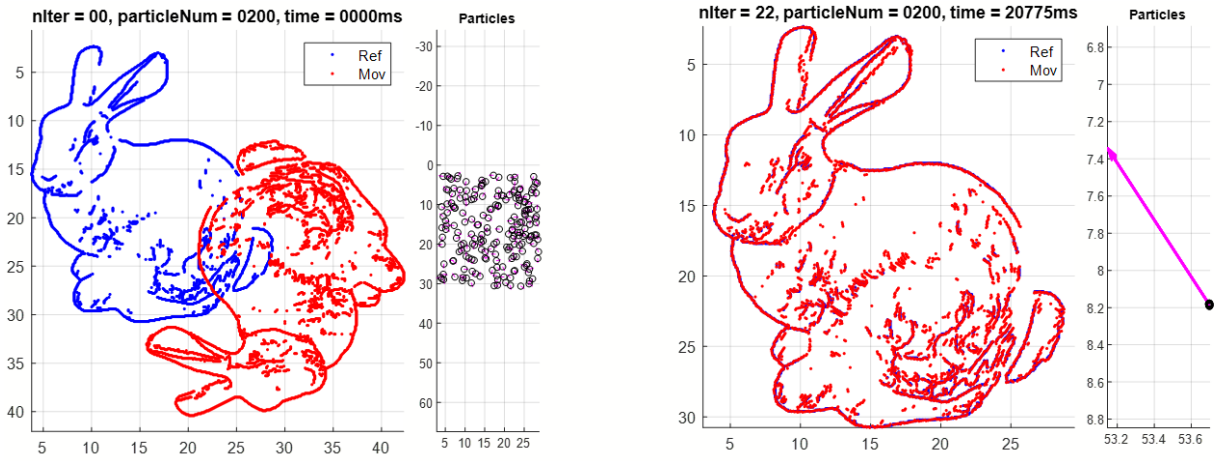

Figure 16: 2D Bunny (No Noise) Before (left) and After (right) PFF Image Registration

6.2 Bunny (2D) Results

The first round of experiments was conducted on the 2D bunny with no noise, medium noise, and high noise. The number of motion updates, *L*, through the ICP propagation model was 7 for both the PF and PFF filters. The PFF used 10 equally spaced steps, $\delta\lambda$.

Figure 17 shows the results of the no noise tests. For the same number of particles, the PFF is on average 6.5% slower than the PF. In achieving 95% alignment, the PFF is 17.7% faster than the PF. The PFF needs 30.0% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 87.3% alignment, which is a reduction in misalignment by 7.7% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 94.6% alignment, which is a reduction in misalignment by 5.4% for the PFF. Table 1 in section 6.6 summarizes these results.

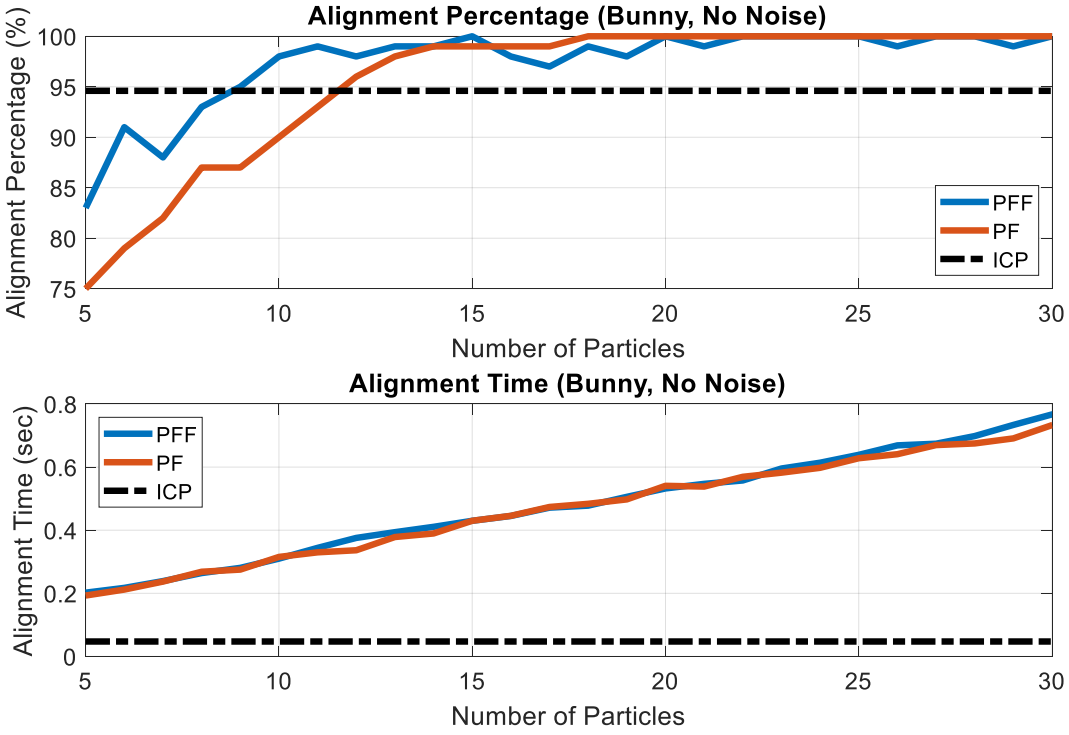

Figure 17: 2D Bunny No Noise Results

Figure 18 shows the results of the medium noise tests, which added WGN with a standard deviation of 5 pixels to the moving image only. For the same number of particles, the PFF is on average 9.6% faster than the PF. In achieving 95% alignment, the PFF is 51.8% faster than the PF. The PFF needs 31.4% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 77.8% alignment, which is a reduction in misalignment by 17.2% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 94.9% alignment, which is a reduction in misalignment by 5.1% for the PFF. Table 2 in section 6.6 summarizes these results.

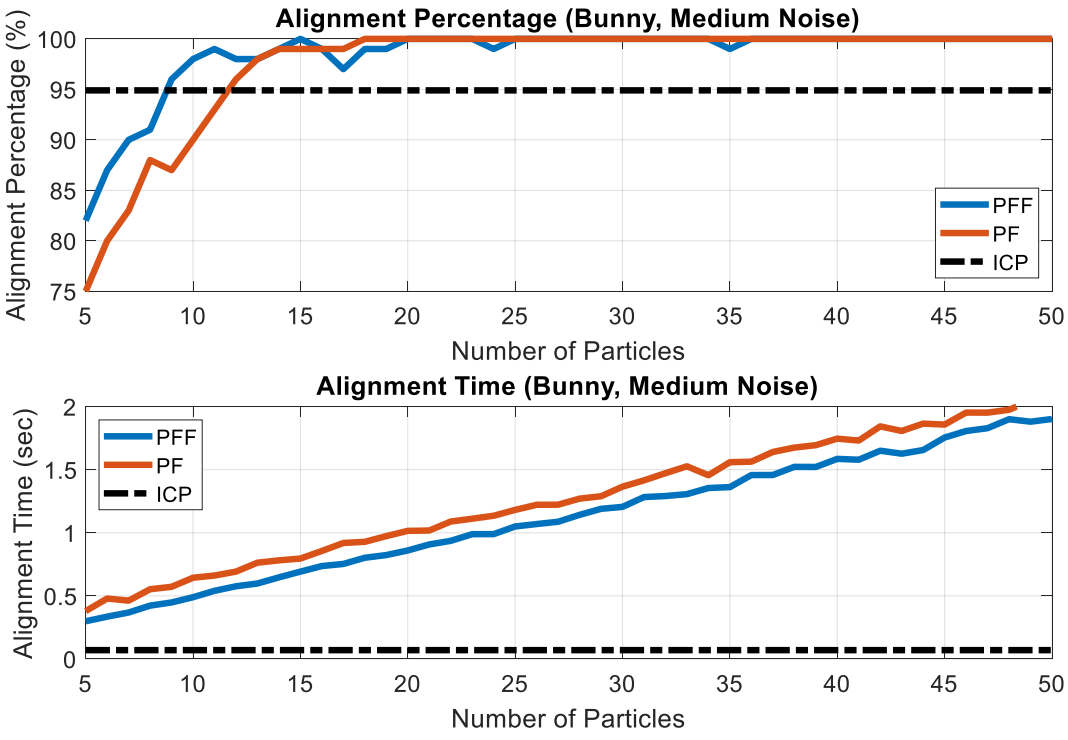

Figure 18: 2D Bunny Medium Noise Results

Figure 19 shows the results of the high noise tests, which added WGN with a standard deviation of 5 pixels to the moving and reference images. For the same number of particles, the PFF is on average 3.0% slower than the PF. In achieving 95% alignment, the PFF is 169.1% faster than the PF. The PFF needs 255.6% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 86.0% alignment, which is a reduction in misalignment by 9.0% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 94.8% alignment, which is a reduction in misalignment by 5.2% for the PFF. Table 3 in section 6.6 summarizes these results.

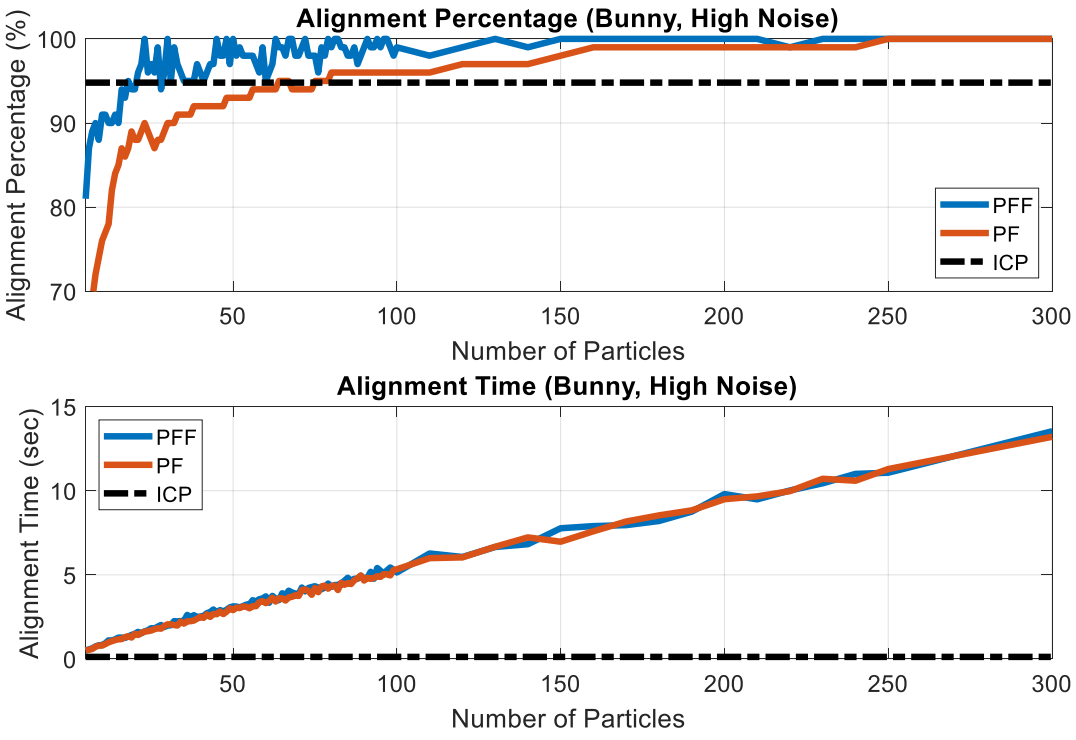

Figure 19: 2D Bunny High Noise Results

6.3 Brain (2D) Results

The second round of experiments was conducted on the 2D brain with no noise, medium noise, and high noise. The number of motion updates, *L*, through the ICP propagation model was 20 for both the PF and PFF filters. The PFF used 10 equally spaced steps, δλ.

Figure 20 shows the results of the no noise tests. For the same number of particles, the PFF is on average 5.8% faster than the PF. In achieving 95% alignment, the PFF is 80.1% faster than the PF. The PFF needs 82.5% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 80.0% alignment, which is a reduction in misalignment by 15.0% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 27.0% alignment, which is a reduction in misalignment by 73.0% for the PFF. Table 1 in section 6.6 summarizes these results.

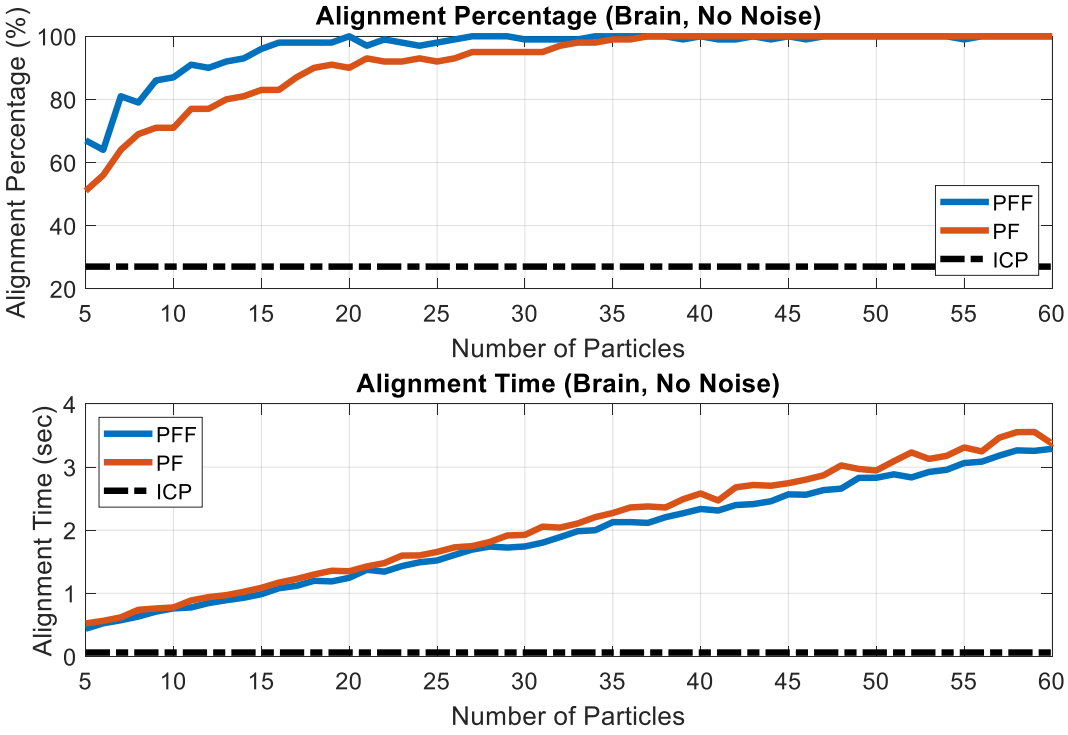

Figure 20: 2D Brain No Noise Results

Figure 21 shows the results of the medium noise tests, which added WGN with a standard deviation of 3 pixels to the moving image only. For the same number of particles, the PFF is on average 21.4% faster than the PF. In achieving 95% alignment, the PFF is 137.5% faster than the PF. The PFF needs 103.8% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 70.1% alignment, which is a reduction in misalignment by 24.9% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 29.0% alignment, which is a reduction in misalignment by 71.0% for the PFF. Table 2 in section 6.6 summarizes these results.

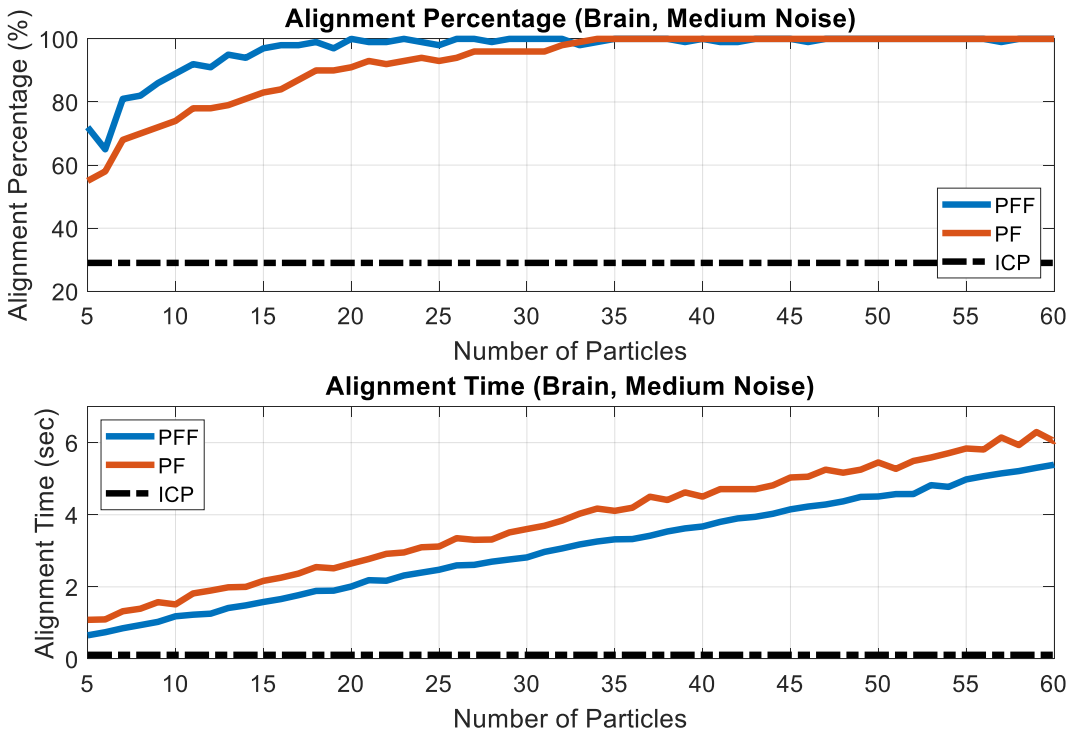

Figure 21: 2D Brain Medium Noise Results

Figure 22 shows the results of the high noise tests, which added WGN with a standard deviation of 3 pixels to the moving and reference images. For the same number of particles, the PFF is on average 6.6% faster than the PF. In achieving 95% alignment, the PFF is 62.7% faster than the PF. The PFF needs 56.7% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 78.2% alignment, which is a reduction in misalignment by 16.8% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 25.0% alignment, which is a reduction in misalignment by 75.0% for the PFF. Table 3 in section 6.6 summarizes these results.

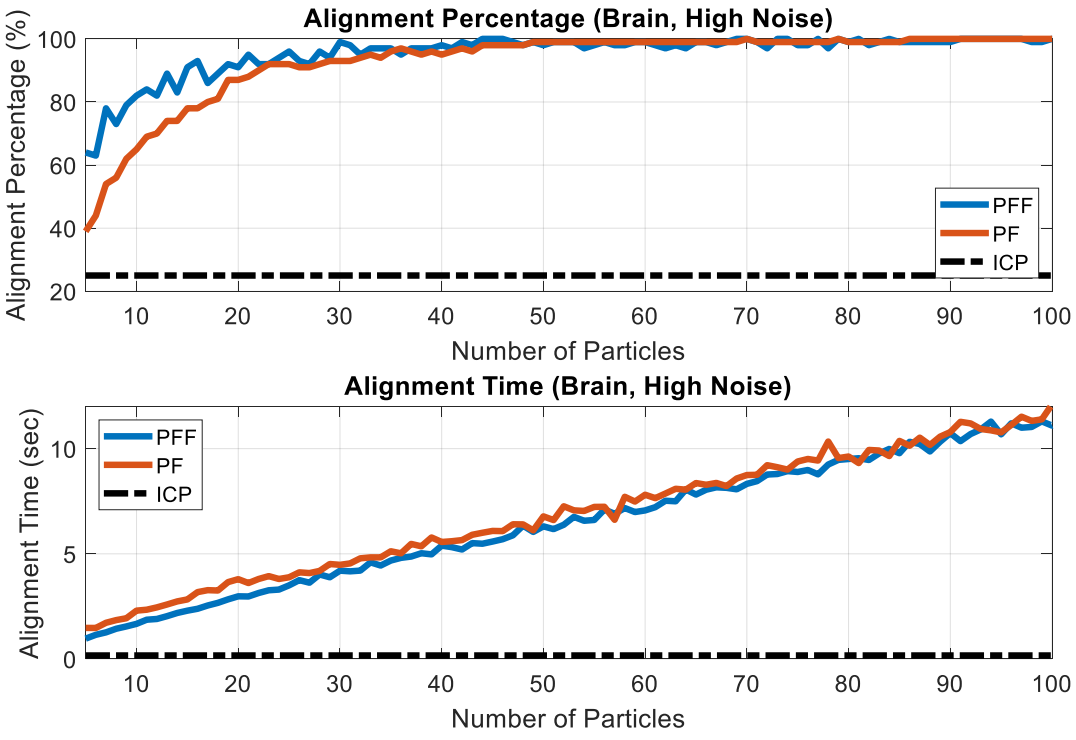

Figure 22: 2D Brain High Noise Results

6.4 Bunny (3D) Results

The third round of experiments was conducted on the 3D bunny with no noise, medium noise, and high noise. The number of motion updates, *L*, through the ICP propagation model was 7 for both the PF and PFF filters. The PFF used 10 equally spaced steps, δλ.

Figure 23 shows the results of the no noise tests. For the same number of particles, the PFF is on average 62.1% faster than the PF. In achieving 95% alignment, the PFF is 87.2% faster than the PF. The PFF needs 20.2% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 77.0% alignment, which is a reduction in misalignment by 18.0% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 73.0% alignment, which is a reduction in misalignment by 27.0% for the PFF. Table 1 in section 6.6 summarizes these results.

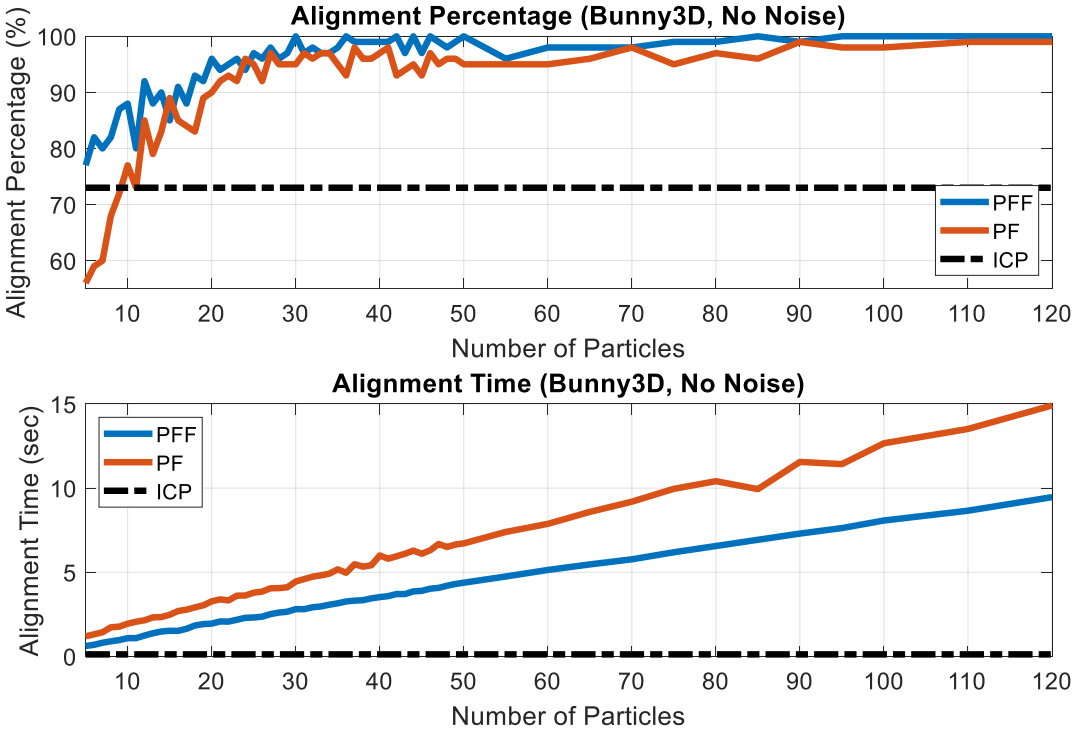

Figure 23: 3D Bunny No Noise Results

Figure 24 shows the results of the medium noise tests, which added WGN with a standard deviation of 5 pixels to the moving image only. For the same number of particles, the PFF is on average 74.1% faster than the PF. In achieving 95% alignment, the PFF is 244.1% faster than the PF. The PFF needs 107.0% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 60.0% alignment, which is a reduction in misalignment by 35.0% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 76.0% alignment, which is a reduction in misalignment by 24.0% for the PFF. Table 2 in section 6.6 summarizes these results.

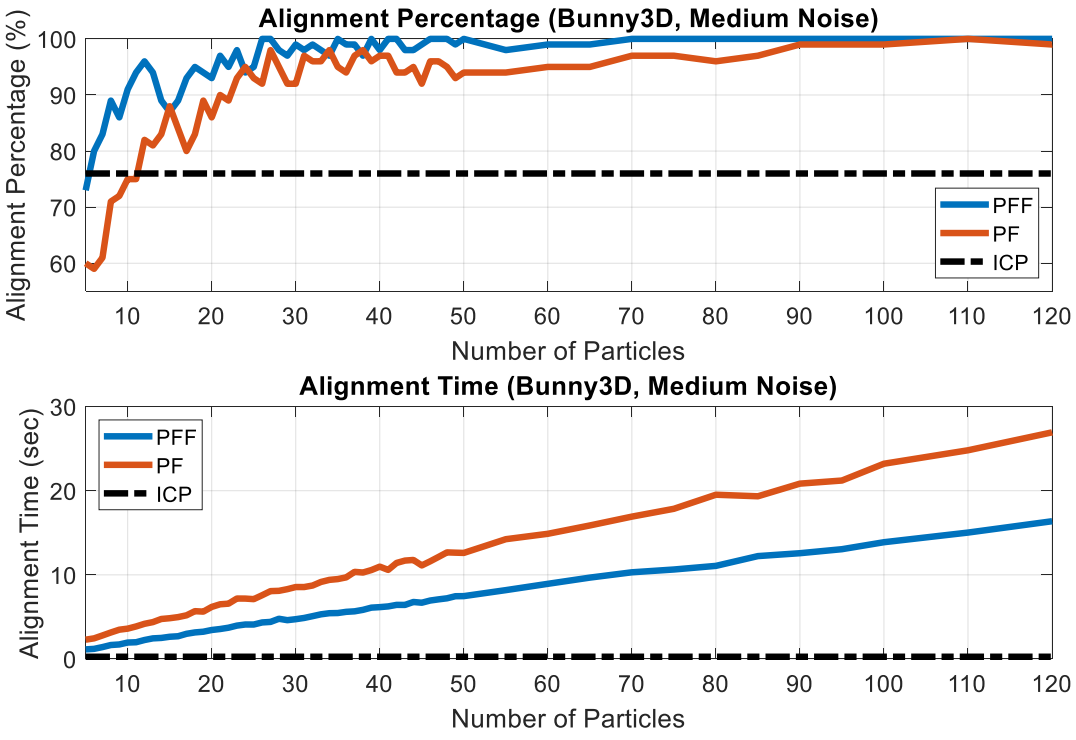

Figure 24: 3D Bunny Medium Noise Results

Figure 25 shows the results of the high noise tests, which added WGN with a standard deviation of 5 pixels to the moving and reference images. For the same number of particles, the PFF is on average 51.0% faster than the PF. In achieving 95% alignment, the PFF is 199.6% faster than the PF. The PFF needs 116.8% less particles than the PF to achieve the same 95% alignment. In the time it takes the PFF to achieve 95% alignment, the PF only achieves 64.6% alignment, which is a reduction in misalignment by 30.4% for the PFF. With enough particles, the PFF achieves 100% alignment, while the ICP only registration achieves 73.0% alignment, which is a reduction in misalignment by 27.0% for the PFF. Table 3 in section 6.6 summarizes these results.

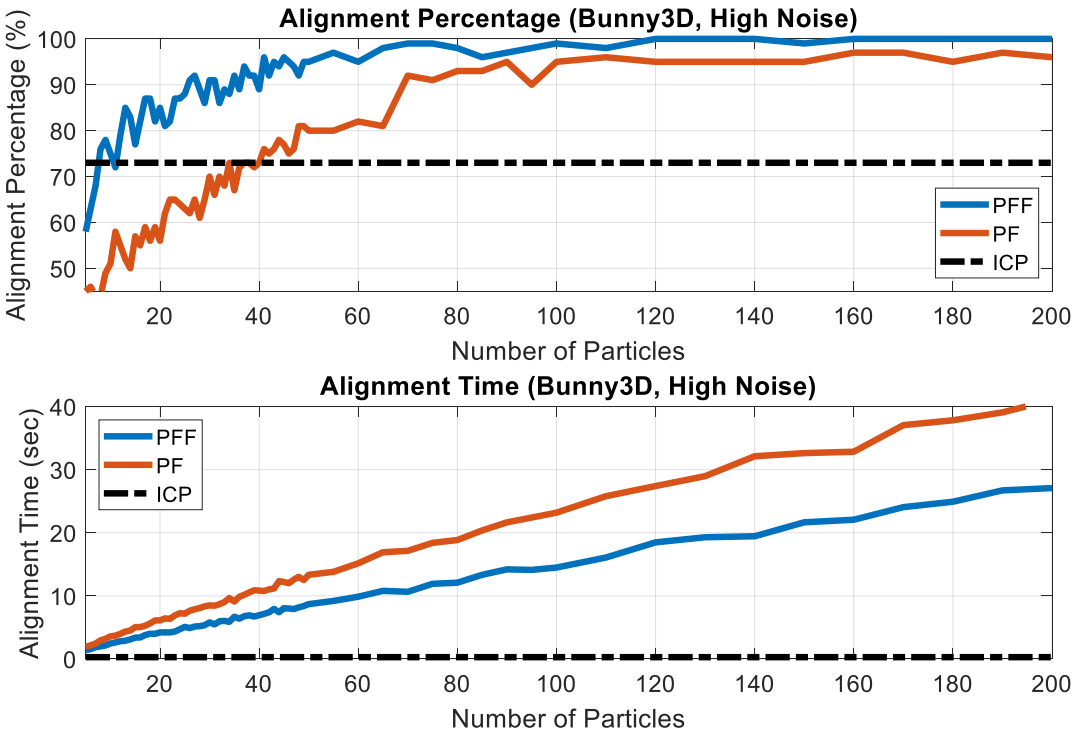

Figure 25: 3D Bunny High Noise Results

6.5 PFF Step Method Results

The secondary set of tests analyzed the PFF in more detail. The number of λ steps ranged from 3 to 100,000 steps, while the number of particles was held constant at 30 particles and 10 particles. The steps were divided either equally or logarithmically between 0 and 1.

Figure 26 shows the alignment percentage as a function of the number of steps for the 2D bunny, 2D brain, and 3D bunny. As expected, more particles (30 versus 10) increase the alignment percentage. However, equal and logarithmic spaced steps produce comparable results. Additionally, increasing the number of steps does not increase alignment percent. The flat performance results for the three image types reveal that the flow of particles is not stiff, but rather flows smoothly from the *a priori* to the *a posteriori*.

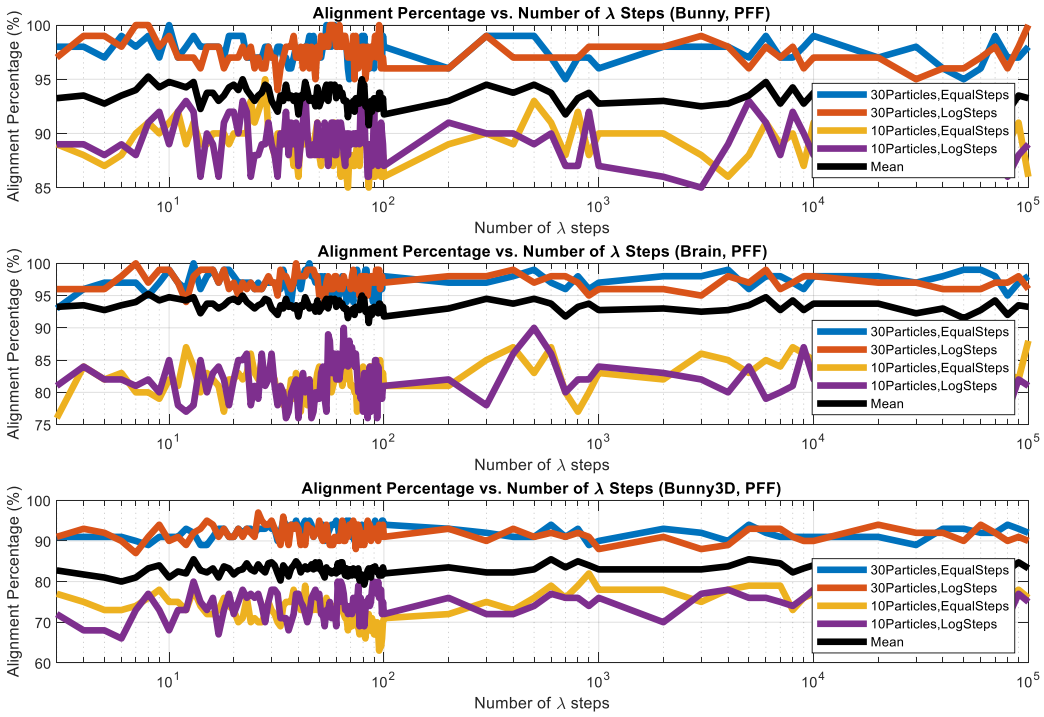

Figure 26: PFF Step Method Results (Alignment Percentage)

Figure 27 shows the alignment time as a function of the number of steps for the 2D bunny, 2D brain, and 3D bunny. As the number of steps increases, the alignment time increases linearly. Therefore, a low number of steps is preferred for PFF image registration from a computational efficiency perspective. Since the step method or number of steps did not impact alignment percentage, the optimal number of steps is less than 100 (i.e. low number of steps). The number of λ steps used in the Monte Carlo analysis of the preceding subsections was 10.

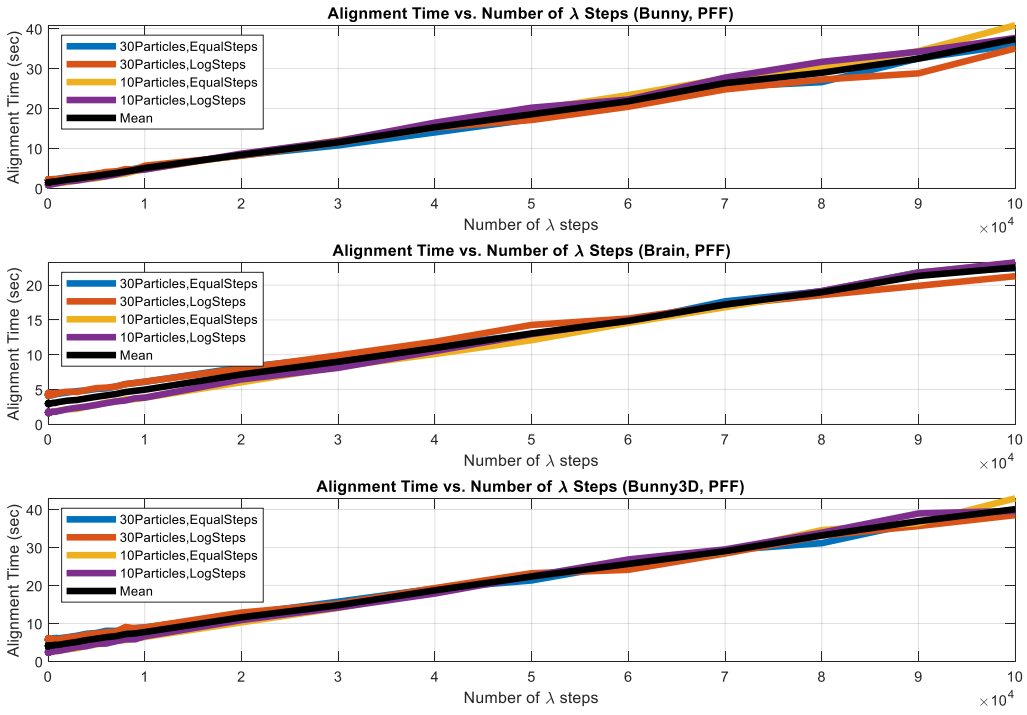

Figure 27: PFF Step Method Results (Alignment Time)

6.6 Performance Summary

Results vary for each image type and noise level; however, in general, the PFF exceeds the PF in both alignment accuracy and alignment time. As expected, the computational efficiency of the PFF over the PF increases as the number of states increases (i.e. going from 2D to 3D). This was predicted based on the distinct advantage of the PFF over the PF regarding the curse of dimensionality. Additionally, higher noise environments and images with more complex features (e.g. 2D brain) favored the PFF over the PF. The PFF Bayesian estimation approach far exceeds the ICP only method regarding alignment percentage; however, the higher accuracy comes at a computational cost. The PFF performed equivalently well with equally spaced and logarithmically spaced steps. A low number of steps (e.g. 10 to 100) is preferred due to the computational burden of processing more steps.

Table 1, Table 2, and Table 3 summarize the testing results for no noise, medium noise, and high noise, respectively. For challenging registration problems, the PFF is as much as 244% faster than the PF in achieving the same alignment percentage. The PFF requires as much as 256% fewer particles than the PF in achieving the same alignment percentage. For the same computational time, the PFF reduces misalignment by as much as 35% compared to the PF. For the same number of particles, the PFF is as much as 74% faster. For all image types and noise levels, the PFF achieves 100% alignment with a sufficient number of particles. The PFF reduces misalignment by as much as 75% compared to the ICP only registration. The innovative PFF image registration algorithms drastically outperform the PF and ICP only image registration methods.

Table 1: Image Registration Results (No Noise)

	2D Bunny	2D Brain	3D Bunny
PFF is X% faster than the PF in achieving 95% alignment	17.7%	80.1%	87.2%
PFF requires X% less particles than the PF in achieving 95% alignment	30.0%	82.5%	20.2%
PF achieves X% alignment in the computational time it takes the PFF to achieve 95% alignment	87.3%	80.0%	77.0%
PF achieves X% less alignment in the computational time it takes the PFF to achieve 95% alignment	7.7%	15.0%	18.0%
PFF is on average X% faster than the PF for an equivalent number of particles	-6.5%	5.8%	62.1%
PFF achieves X% alignment	100.0%	100.0%	100.0%
ICP only method achieves X% alignment	94.6%	27.0%	73.0%
PFF reduces misalignment by X% compared to ICP only	5.4%	73.0%	27.0%

*Key: **Green** favors the PFF, **Red** favors the PF, and **Black** is informative text only

Table 2: Image Registration Results (Medium Noise)

	2D Bunny	2D Brain	3D Bunny
PFF is X% faster than the PF in achieving 95% alignment	51.8%	137.5%	244.1%
PFF requires X% less particles than the PF in achieving 95% alignment	31.4%	103.8%	107.0%
PF achieves X% alignment in the computational time it takes the PFF to achieve 95% alignment	77.8%	70.1%	60.0%
PF achieves X% less alignment in the computational time it takes the PFF to achieve 95% alignment	17.2%	24.9%	35.0%
PFF is on average X% faster than the PF for an equivalent number of particles	9.6%	21.4%	74.1%
PFF achieves X% alignment	100.0%	100.0%	100.0%
ICP only method achieves X% alignment	94.9%	29.0%	76.0%
PFF reduces misalignment by X% compared to ICP only	5.1%	71.0%	24.0%

*Key: **Green** favors the PFF, **Red** favors the PF, and **Black** is informative text only

Table 3: Image Registration Results (High Noise)

	2D Bunny	2D Brain	3D Bunny
PFF is X% faster than the PF in achieving 95% alignment	169.1%	62.7%	199.6%
PFF requires X% less particles than the PF in achieving 95% alignment	255.6%	56.7%	116.8%
PF achieves X% alignment in the computational time it takes the PFF to achieve 95% alignment	86.0%	78.2%	64.6%
PF achieves X% less alignment in the computational time it takes the PFF to achieve 95% alignment	9.0%	16.8%	30.4%
PFF is on average X% faster than the PF for an equivalent number of particles	-3.0%	6.6%	51.0%
PFF achieves X% alignment	100.0%	100.0%	100.0%
ICP only method achieves X% alignment	94.8%	25.0%	73.0%
PFF reduces misalignment by X% compared to ICP only	5.2%	75.0%	27.0%

*Key: **Green** favors the PFF, **Red** favors the PF, and **Black** is informative text only

7 Conclusion

This thesis project describes an innovative registration algorithm using the particle flow filter (PFF). This paper began by introducing image registration, the particle filter (PF), and the PFF. Next, point-set registration via the PFF was presented. Others have approached image registration as a Bayesian filtering problem; however, none have used the PFF. The innovative aspect of this project is that image registration is viewed as a Bayesian filtering problem and solved with the PFF. The PFF provides distinct advantages over other Bayesian filtering methods. The PFF is computationally more efficient than other methods such as the better-known PF. Additionally, the PFF is not constrained to the highly restrictive unimodal, linear, and Gaussian assumptions of many Bayesian filters such as the Kalman filter. The PFF works for any probability density function. Regarding image registration, the Bayesian filtering framework provides significant advantages such as increased robustness to noise, outliers, and local minima.

Finally, the innovative particle flow filter algorithms were implemented using MATLAB. Both 2D and 3D rigid body point-set registration was conducted using the Gromov particle flow filter variant. Additionally, the particle filter method proposed by [1] and iterative closest point algorithms were implemented for comparison. All three registration techniques were tested with a high degree of initial misalignment and noise. For the same alignment accuracy, the new particle flow filter algorithms were 244% faster than the particle filter for certain challenging problems. For the same alignment time, the particle flow filter reduced misalignment by as much as 35% compared to the particle filter. The particle flow filter achieved 100% alignment with enough particles, and reduced misalignment by as much as 75% over that of iterative closest

point. These results demonstrate that image registration via the particle flow filter significantly outperforms the particle filter and iterative closest point algorithms in the presence of noise and a high degree of initial misalignment. Although applicable to affine and elastic registration, this project focused on 2D and 3D rigid body image registration. Future areas of research for image registration using the particle flow filter include deformable registration and GPU parallelization.

References

[1] R. Sandhu, S. Dambreville, and A. Tannenbaum, “Point Set Registration via Particle Filtering and Stochastic Dynamics,” published in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 32, No. 8, Aug. 2010, pp. 1459-1473, available through *National Institutes of Health*, Author Manuscript, pp. 1-35.

[2] M. Moghari, and P. Abolmaesumi, “Point-Based Rigid-Body Registration Using an Unscented Kalman Filter,” in *IEEE Transactions on Medical Imaging,* vol. 26, no. 12, pp. 1708-1728, Dec. 2007.

[3] E. Arce-Santana, D. Campos-Delgado, and A. Alba, “Affine Image Registration Guided by Particle Filter,” *IET Image Processing*, 2012, Vol. 6, Iss. 5, pp. 455-462.

[4] B. Zitova and J. Flusser, “Image registration methods: a survey,” Elsevier *Image and Vision Computing 21*, 2003, pp. 977-1000.

[5] K. Kulhavy. Registrator Demo 2 [Photograph]. Available: https://commons.wikimedia.org/wiki/File:Registrator_Demo2.png

[6] A. Keszei, B. Berkels, and T. Deserno, “Survey of Non-Rigid Registration Tools in Medicine,” *Journal of Digital Imaging*, 2017, pp. 102-116.

[7] N. Sladoje. (2016). Image Registration [Slides]. Available: http://www.it.uu.se/edu/course/homepage/bild2/v17/schedule/Registration_2016.PDF

[8] R. Sandhu, S. Dambreville, and A. Tannenbaum, “Particle Filtering for Registration of 2D and 3D Point Sets with Stochastic Dynamics,” *2008 IEEE Conference on Computer Vision and Pattern Recognition,* Anchorage, AK, 2008, pp. 1-8.

[9] K. Arun, T. Huang and S. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698-700, Sept. 1987.

[10] F. Gustafsson, “Particle Filter Theory and Practice with Positioning Applications,” *IEEE Aerospace and Electronic Systems Magazine,* vol 25, no. 7, pp. 53-82, July 2010.

[11] S. Choi, P. Willett, F. Daum, and J. Huang, “Discussion and Application of the Homotopy Filter,” *Signal Processing, Sensor Fusion, and Target Recognition XX,* edited by I. Kadar, *Proc. of SPIE,* Vol 8050, 2011, pp. 1-12.

[12] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," in *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174-188, Feb. 2002.

[13] P. Maybeck, *Stochastic Models, Estimation, and Control, Volume 1.* New York, NY: Academic Press, 1979.

[14] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," Department of Computer Science, University of North Carolina at Chapel Hill, TR 95-041, 2006.

[15] F. Daum and J. Huang, "Particle Degeneracy: Root Cause and Solution," *Signal Processing, Sensor Fusion, and Target Recognition XIX,* edited by I. Kadar, *Proc. of SPIE,* Vol 8050, 2011, pp. 1-11.

[16] V. Jilkov, J. Wu, and H. Chen, "Performance Comparison of GPU-Accelerated Particle Flow and Particle Filters," in *16th International Conference on Information Fusion*, Istanbul, Turkey, 2013, pp. 1095-1102.

[17] D. Crouse, "Particle Flow Filters: Biases and Bias Avoidance," *22nd International Conference on Information Fusion*, Ottawa, Canada, 2019, pp. 1-8.

[18] F. Daum, J. Huang, and A. Noushin, "New Theory and numerical results for Gromov's Method for stochastic particle flow filter," *2018 21st International Conference on Information Fusion (FUSION)*, Cambridge, 2018, pp. 108-115.

[19] F. Daum, J. Huang, and A. Noushin, "Exact Particle Flow for Nonlinear Filters," *Signal Processing, Sensor Fusion, and Target Recognition XIX,* edited by I. Kadar, *Proc. of SPIE,* Vol 7697, 2010, pp. 1-19.

[20] H. Risken, *The Fokker-Planck Equation: Methods of Solution and Application*. Berlin, Germany: Springer-Verlag, 1984.

[21] F. Daum and J. Huang, "Seven Dubious Methods to Mitigate Stiffness in Particle Flow with Non zero Diffusion for Nonlinear Filters, Bayesian Decisions and Transport," *Signal and Data Processing of Small Targets,* edited by O. Drummon, *Proc. of SPIE,* Vol 9092, 2014, pp. 1-11.

[22] F. Daum and J. Huang, "Exact Particle Flow for Nonlinear Filters: Seventeen Dubious Solutions to a First Order Linear Underdetermined PDE," *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers,* Pacific Grove, CA, 2010, pp.64-71.

[23] F. Daum, J. Huang, and A. Noushin. (2018). Gromov's method for stochastic particle flow filters [Lecture & Slides]. Available: https://www.youtube.com/watch?v=vqJGB47XoeY

[24] D. Crouse and C. Lewis, “Consideration of Particle Flow Filter Implementations and Biases,” Naval Research Laboratory Memo, 2019, pp. 1-17.

[25] Y. Li, L. Zhao, and M. Coates, “Particle flow for particle filtering,” *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Shanghai, 2016, pp. 3979-3983.

[26] M. Mallick and B. Sindhu, “Critical Analysis of the Particle Flow Filter,” *2015 International Conference on Control, Automation, and Information Sciences (ICCAIS)*, Changshu, 2015, pp. 512-517.

[27] D. Wang, *Vanilla_PF_ICP* (2020), available: https://github.com/DrGabor/Vanilla_PF_ICP

[28] Stanford Graphics Archive (2020), available: http://graphics.stanford.edu/data/

Curriculum Vitae

Stephen Porter received the Bachelor of Science degree in Electrical Engineering from the United States Air Force Academy in 2007. Upon graduation, he was an officer and radar analyst in the United States Air Force. He currently works for Raytheon Technologies as a system analyst and algorithm developer. His primary areas of expertise are radar performance analysis, signal processing, and algorithm development. His research interests include synthetic aperture radar, target classification & discrimination, and image registration. This thesis project completes his Master of Science degree in Electrical Engineering with an emphasis in digital signal processing from Johns Hopkins University. He is a private pilot and enjoys spending time with his family.